AFRL-AFOSR-UK-TR-2011-0027

COMMUNICATION AND DISTRIBUTED CONTROL IN MULTI-AGENT SYSTEMS

Angelo Cangelosi
Fabio Ruini

University of Plymouth
School of Computing Comms Electronics
Drake Circus
Plymouth, United Kingdom PL4 8AA

EOARD GRANT 07-3075

August 2011

Final Report for 15 May 2007 to 15 May 2011

Air Force Research Laboratory
Air Force Office of Scientific Research
European Office of Aerospace Research and Development
Unit 4515 Box 14, APO AE 09421

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
03-08-2011	Final Report	15 May 2007 – 15 May 2011

4. TITLE AND SUBTITLE

COMMUNICATION AND DISTRIBUTED CONTROL IN MULTI-AGENT SYSTEMS

5a. CONTRACT NUMBER

FA8655-07-1-3075

5b. GRANT NUMBER

Grant 07-3075

5c. PROGRAM ELEMENT NUMBER

61102F

6. AUTHOR(S)

Professor Angelo Cangelosi
Fabio Ruini

5d. PROJECT NUMBER

5d. TASK NUMBER

5e. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Plymouth
School of Computing Comms Electronics
Drake Circus
Plymouth, United Kingdom PL4 8AA

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

EOARD
Unit 4515 BOX 14
APO AE 09421

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/AFOSR/RSW (EOARD)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

AFRL-AFOSR-UK-TR-2011-0027

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This project has focused on the design of distributed autonomous controllers for collective behavior of Micro-unmanned Aerial Vehicles (MAVs). Two alternative approaches to this topic are introduced: one based upon the Evolutionary Robotics (ER) paradigm, the other one upon flocking principles. Three computer simulators have been developed in order to carry out the required experiments, all of them having their focus on the modeling of fixed-wing aircraft flight dynamics. The employment of fixed-wing aircraft rather than the omni-directional robots typically employed in collective robotics significantly increases the complexity of the challenges that an autonomous controller has to face. This is mostly due to the strict motion constraints associated with fixed-wing platforms that require a high degree of accuracy by the controller. Concerning the ER approach, the experimental setups elaborated have resulted in controllers evolved in simulation with the following capabilities: (1) navigation across unknown environments, (2) obstacle avoidance, (3) tracking of a moving target, and (4) execution of cooperative and coordinated behaviors based on implicit communication strategies. The design methodology based upon flocking principles has involved tests on computer simulations and subsequent experimentation on real-world robotic platforms. A customized implementation of Reynolds' flocking algorithm has been developed and successfully validated through flight tests performed with the swinglet MAV. It has been notably demonstrated how the Evolutionary Robotics approach could be successfully extended to the domain of fixed-wing aerial robotics, which has never received a great deal of attention in the past. The investigations performed have also shown that complex real physics-based computer simulators are not a compulsory requirement when approaching the domain of aerial robotics, as long as proper autopilot systems (taking care of the "reality gap" issue) are used on the real robots.

15. SUBJECT TERMS

EOARD, Multi Agent Systems, Machine Learning, Cognition, Language processing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18, NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			JAMES LAWTON Ph. D.
UNCLAS	UNCLAS	UNCLAS	SAR	63	19b. TELEPHONE NUMBER *(Include area code)* +44 (0)1895 616187

EOARD GRANT 073075

COMMUNICATION AND DISTRIBUTED CONTROL IN MULTI-AGENT SYSTEMS

Principal Investigator: Professor Angelo Cangelosi
PhD Student: Mr Fabio Ruini

Final Report

ABSTRACT

The goal of this project has been to develop new Multi-Agent Systems (MAS) models of communication in collaborative tasks to investigate the role of explicit communication in distributed control. This project has focused on the role of explicit symbolic communication between software agents in distributed control tasks. Compared to centralized control, where a central controller is responsible for the (pre)planning, task-assignment and supervision of the coordination task, in distributed control systems intelligent autonomous, or semi-autonomous, agents are capable of sensing, acting, cognition and communication and together contribute to the task solution. These network-centric systems only require partial interaction with other agents, and may necessitate simpler architectures and individual resource requirements as knowledge is distributed in the population. The significant advantages of this approach are that the system is more robust, adaptive and fault tolerant since there is no critical reliance on any specific individual, and that decentralization results in increased reliability, safety and speed of response. In addition, distributed approaches have the benefit of not requiring the full pre-planning of the cooperative strategy. Adaptive solutions can emerge run-time through the interaction between autonomous individuals and from the task and environment requirements which might not be fully accessible (known) at the beginning of the problem.

New studies on the role of explicit communication in MAS have many theoretical and technological implications. First, agents that are allowed to communicate explicitly during the execution of collaborative task might benefit from the exchange of information regarding properties of the task being processed. Such explicit communication systems do not have to be defined a priory by the human designer, but can autonomously emerge from social interaction between agents. A second advantage of studying symbolic communication concerns the development of human-centered systems and hybrid human/agent/robot systems. This will support the reconciliation of human decision making schemes with machine performance and intelligent agents, keeping the human in the loop. Finally, the post-hoc analysis of the communication systems developed by the agents (with or without interactions with human users) can give meaningful insights on the optimal strategies upon which the distributed control strategy is based. This can be also used for the design and improvement of human-centered distributed control systems.

Table of Contents

EOARD GRANT 073075

Communication and Distributed Control in Multi-Agent Systems
http://www.tech.plym.ac.uk/soc/research/ABC/plymav/

Principal Investigator: Professor Angelo Cangelosi
PhD Student employed on the project: Mr Fabio Ruini

Second Interim Report (Year 1, Month 12)

Introduction

This report will first provide brief summary of the work carried out during the first six months of the project, and will then describe in detail the two main experiments carried out during the second semester. These new experiments respectively employ a target able to move (trying to escape from the approaching MAVs) and a more "robust" target that requires two sequential hits in order to be destroyed. The report will also contain sections on the description of some additional studies on the genetic algorithm parameters, in order to optimize its convergence speed. We will discuss as well new experiments where we tried to extend the validity of the developed model, applying the same basic principles on different simulated environments/locations. We will also address some of the criticisms raised during the presentations of this ongoing work at scientific conferences and workshops. Finally we will describe the plans for future work that mainly concern the extension of the previous model to a three-dimensional simulator, that will act as the base platform on which the future communication-related experiments will be carried out. Full details on the experiments carried out and the software platform are available in the project website: http://www.tech.plym.ac.uk/soc/research/ABC/plymav/

Summary of previous work (experiments A and B)

The main goal of this project is to develop embodied neural network controllers for MAV (Micro-unmanned Aerial Vehicles) swarms employing Evolutionary Robotics methodologies [1]. Each swarm's member will have to be able to navigate autonomously through an unknown environment, relying only on a specified subset of information gathered through its sensors. Communicating with its teammates in order to perform tasks requiring coordination and cooperation will be studied in the second part of the project.

The approach we have chosen to follow falls into the so-called "reactive strategies" category as classified by Richards et al. [2]. The main difference with respect to standard reactive strategy approach mainly consists in the employment of a neural network controller instead of a properly defined decision tree. In both cases the training[1] of the autonomous controllers is based on evolutionary algorithm and the use of computer simulator experiments.

With the first experiments carried out in the first semester (see [3][4]) we outlined the general requirements for a simulator of this kind, and identified the minimal sensors requirements for allowing MAVs to perform navigation tasks both inside plain (experiment A) and obstacle-full (experiment B) environments.

[1] We are using the term "training" in a general sense, even if it is not the most appropriate to use. We are not referring in fact to learning algorithms (e.g. back-propagation) for which the term "training" is typically employed, but to a search of the most appropriate set of connection weights and biases pursued through evolutionary algorithms.

Figure 1 - The simulated environment (based on a map of Canary Wharf, London obtained through Google Earth) used for experiment B. Its size is 710 x 760 pixels, corresponding to 630 x 675 meters

The experimental conditions are quite simple. A swarm is composed by four MAVs, each endowed with its own neural network controller, identical to the ones of its teammates. At the beginning of a test, an "enemy" target is deployed somewhere inside the environment. The test environment consists of the Canary Warf financial district in London. Starting from the four environment corners, the MAVs have to fly toward the target attempting to eliminate it. In order to neutralize the target, one of the MAVs needs to perform a detonation when it is close enough to it (2.2 meters or less). A test ends when the target has been destroyed or no MAVs are still living. A MAV will die if it performs a detonation, if it attempts to exit from the environment's boundaries, if it collides against a teammate, if it runs out of energy (every movement it does costs a certain amount of energy), and, in experiment B, if it crashes against a building.

Each MAV is able to perceive the distance between itself and the target, as well as the angle that separates the two agents (based on the current MAV's facing direction)[2]. In experiment B the MAVs are also endowed with three ultra-sonic sensors, capable to detect the presence of (and the distance from) an obstacle, which could be the target, a teammate or a building. The neural network's output layer consists of two neurons. One output unit controls the MAV's steering direction (+/-20 degrees in the time unit); the other output unit is a Boolean neuron that, when it turns to 1, causes the MAV to carry out the detonation. The fact that we are simulating an aircraft-like motion implies the constraint, for the MAVs, of being always on movement. The speed is instead assumed as constant.

The evolution toward a controller able to perform the desired task is made possible by a genetic algorithm. An initial population of 100 swarms is created with randomly assigned connection weights and biases. Each swarm is tested four times with the target deployed in randomly chosen positions. At the end of each generation the 20 individuals that has performed the best scores according to the fitness formula are selected for reproduction. Each swarm generates 5 copies of its genome, on which the mutation operator is then applied. The resulting 100 individuals will constitute the new population at the next generation. The evolutionary process lasts for 2,500 generations and it is repeated 10 times with the results of all the runs averaged in order to obtain more reliable data.

[2] These assumptions are supported by the idea of a satellite system that continuously monitors the target and broadcast the information about its coordinates to the MAVs. In this way the MAVs - equipped with a GPS receiver - can easily calculate their distance from the target matching the two data sources received. Then a simple compass can easily allow the MAVs to determine also the relative direction on which the target is.

The fitness formulas used are the following, respectively for experiments A and B:

$$fitness = -\alpha + (\frac{\beta}{50}) + (\sigma * 50) + (\phi * 5)$$

$$fitness = -\alpha + (\frac{\beta}{50}) + (\sigma * 50) + (\phi * 10)$$

where: α is the average distance between the target and the swarm's member detonated closest to it, calculated during the four tests; β is the average amount of energy retained by the MAV detonated closest to the target; σ is the number of tests concluded by the given swarms with the elimination of the target; Φ is the total number of swarm's members remained alive after the 4 tests (maximum 3 MAVs * 4 tests = 12). The only difference between the two fitness formulas consists in the part of the equation involving the Φ parameter. In the second experiments it is multiplied by 10 – twice as before - in order to facilitate the evolution of the obstacle-avoidance behavior.

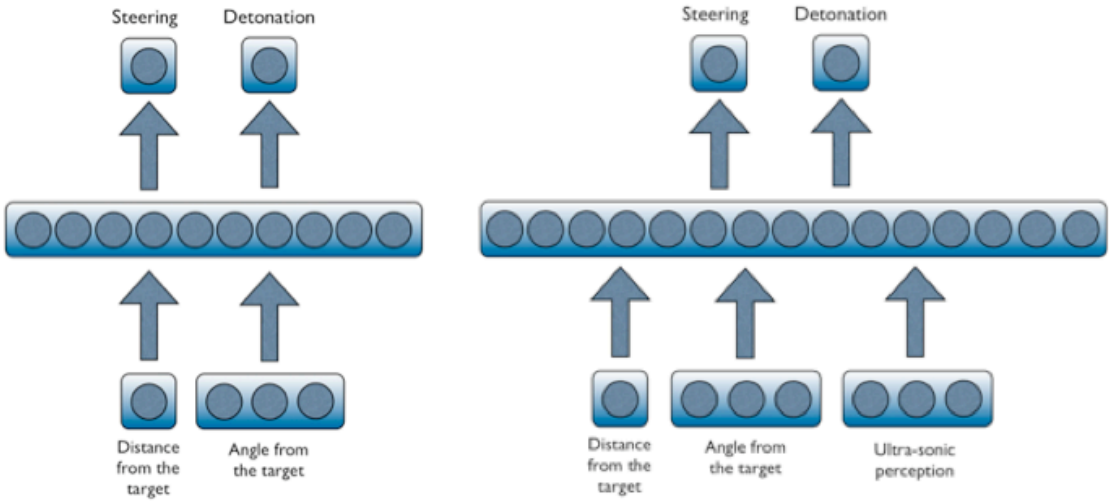


Figure 2 - The neural network architectures employed for experiment A (left) and B (right)

The results show how the set up elaborated could lead quite easily to the proper evolution of the desired behaviours. At the end of the evolution, on average we have the 93.46% of tests successfully concluded for experiment A and 87.18% for experiment B (see Figure 3).

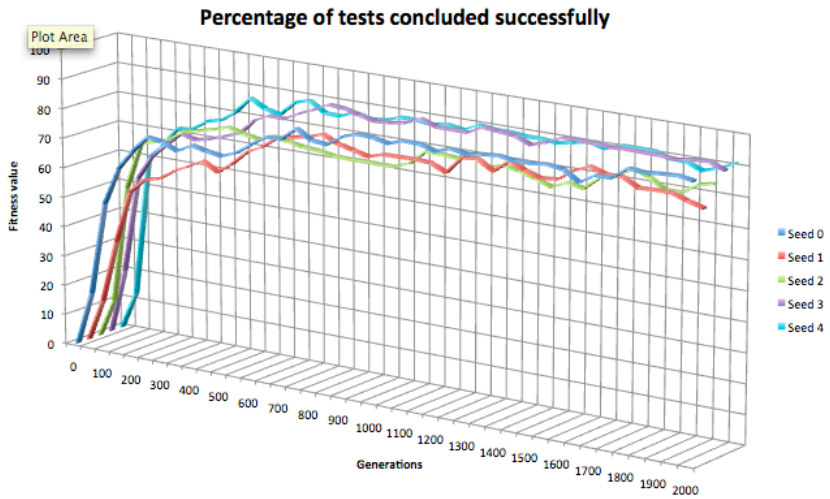


Figure 3 - Percentage of tests concluded with the elimination of the target for experiment B

New experiment C: Task with movable target

In this new experimental scenario, the target is able to detect a MAV approaching it. This new property of the target was introduced to increase the complexity of the task and tets the robustness of the model. During each time step, if a MAV is closer than 15 meters to the target, the latter can detect the aircraft with probability .5. In the event of MAV detection, the target will then move away from the MAV in order to maximize its distance from the approaching aircraft. The target will remain in "MAV detected mode", and will keep moving away during every step, until the aircraft will die or the distance between the two agents will be over the 38 meters threshold.

Five simulations have been carried out where we varied the escaping speed of the target. Given the MAVs' flying speed of 55 km/h, the target speeds in the various simulations respectively correspond to 27.5 (Simulation C1: target speed = MAV speed/2), 18.33 (Simulation C2: target speed = MAV speed/3), 13.75 (Simulation C3: target speed = MAV speed/4), 11 (Simulation C4: target speed = MAV speed/5) and 9.16 (Simulation C5: target speed = MAV speed/6) km/h.

The results are summarized into the following table:

Simulation	Av. fitness	Max fitness	Av. success %	Av. distance from the target (px)	Min distance from the target (px)
C1	138.93	395.18	54.58	92.94	1.34
C2	198.08	409.67	78.28	107	1.09
C3	250.68	411.74	81.89	72.47	0.95
C4	258.72	409.9	83.3	70.03	0.98
C5	242.42	413.05	84.06	95.53	0.81

Table 1 - Summary of the results obtained from the various simulations of experiment C

Observing the outcome of these simulations - particularly with regard to the average fitness - we can easily identify a kind of threshold. Simulations C3, C4 and C5 seem to perform equally well according to the various parameters measured. Simulation C2 produces a significantly worse performance for the average fitness, but performs quite well if we take into account the maximum fitness and the average percentage of tests concluded successfully. In simulation C1 the performance is rather poor.

Comparing these results with the ones obtained before using a static target, we can notice a general performance decrement. This was largely expected, since the new task is more difficult than the previous one. As clearly shown in Figure 4, the difference is mainly concentrated on the average values, while the maximum ones (i.e., the best individuals/controllers within a certain generation) tend to reach similar level of performance.

The main conclusion of experiment C is that the previous algorithm setup can evolve MAV controller able to navigate through unknown environments and autonomously reach and destroy a target able to move away from them. The only constraint is that the target should not be able to move faster than one third of the MAVs' speed. This is quite a reasonable assumption if we suppose that the target is not a vehicle, but a person. Considering that the moving speed of an average person moving in crowded environment could approximated be 4-7 km/h while walking, and 15-20

km/h while running, we might argue that the evolved controllers are able to accomplish their task with a good degree of confidence even against a movable target[3].

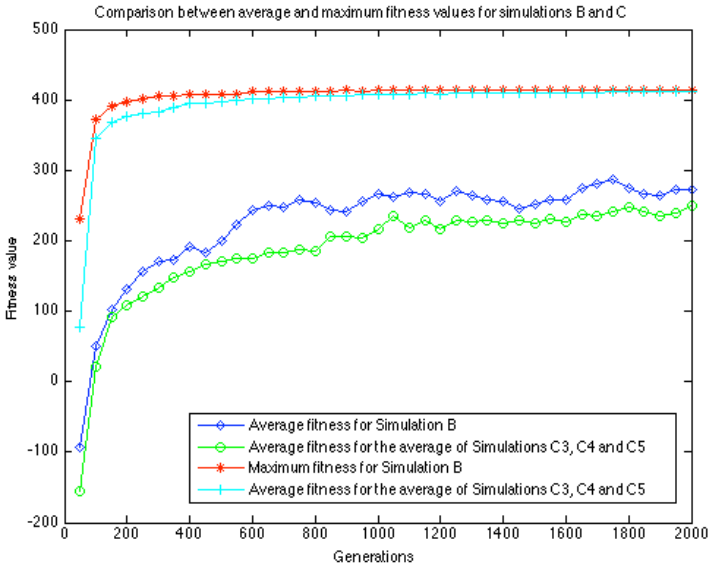


Figure 4 - Average and maximum fitness compared for simulations B and C3, C4 and C5 (the last three averaged)

New Experiment D: Cooperative task

The setup labeled as experiment D adds the constraint of requiring two MAVs to detonate at the same time (i.e., within limited max number of time-steps apart from each other, since the simulation works in discrete time steps) against the target in order to neutralize it. The target begins each training epoch with the assigned status of "intact". When it happens that one of the MAVs detonates close enough to it (i.e., the same situation that in the previous setups would have provoked the elimination of the target), the target's status switches to "damaged". If a second MAV manages to detonate close enough to the target while this is still in the "damaged" mode, the target will be eliminated. Otherwise, after 10 time-steps of "damaged" state, the target will restore its original "intact" condition.

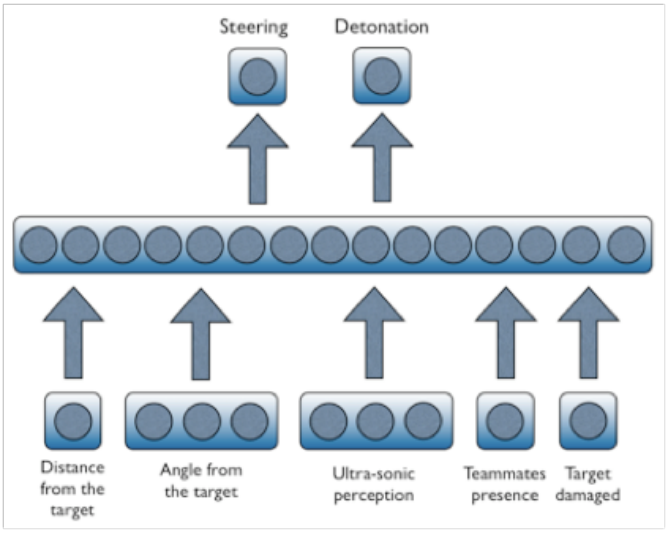


Figure 5 - The neural network architecture used for experiment D

[3] Consider that a typical MAV platform, as could be the Aerovironment's WASP III, is able to reach a speed of 65 km/h (http://www.avinc.com/downloads/Wasp_III.pdf). One third of this speed roughly corresponds to 21.5 km/h, which is a value comparable to the 15-20 km/h suggested as the maximum speed for an average person running.

In order to make the MAVs able to accomplish this task, we have provided them with the capability of gathering additional information from the environment. Each member of the swarm is able to detect both the status of the target (healthy or damaged) and the presence of a teammate within a certain distance. This information is given in input to the neural controller through two additional Boolean neurons. These two neurons implement a kind of logic OR. In order to detonate, a MAV needs to know that there is a teammate close to it, or that the target has recently been damaged (i.e., it is currently on the "damaged" status), or that both conditions (closeness and damaged) are true. Few neurons have been added to the hidden layer as well, in order to make the neural network able to cope with the increased amount of information collected from the environment.

The fitness formula was also modified in order to let the new desired behavior evolve. We have now introduced the concepts of "target approached" and "target damaged". At the end of a test, we define the target as "approached" if at least one MAV has detonated within a 56 meters range from it. The target is considered "damaged" instead if at least one MAV has managed to detonate against it. These modifications tend to recreate what we could call an incremental evolutionary process (though in a different way from the strategy used by Barlow and colleagues [5]). The MAVs initially learn how to perform the simplest sub-task (approaching the target) and then progressively move toward the more complicated sub-tasks (damaging the target and neutralizing the target), which in turns make the accomplishment of the overall task possible.

Putting all together, the new fitness formula is:

$$fitness = (\gamma * \frac{\omega}{4}) + (\eta * \frac{\omega}{2}) + (\lambda * \omega) + (\phi * 10) + (\frac{\beta}{50})$$

where: Υ is the number of tests concluded with at least one MAV "approaching" the target; η is the number of tests concluded with at least one MAV "damaging" the target; λ is the number of tests concluded successfully and $\omega = 50$ (since this is just a parameter chosen in order to assign a different specific weight to Υ, η and λ). Parameters Φ and β have a similar meaning to the one they had before, as they respectively represent the total number of swarm's members survived at the end of the all tests (consider that this number has been extended from 4 to 12) and the average amount of energy retained by the MAV that had eventually neutralized the target. Also consider that Φ is still multiplied for 10 as in the previous Simulations, so the balancing of the fitness function has changed since before.

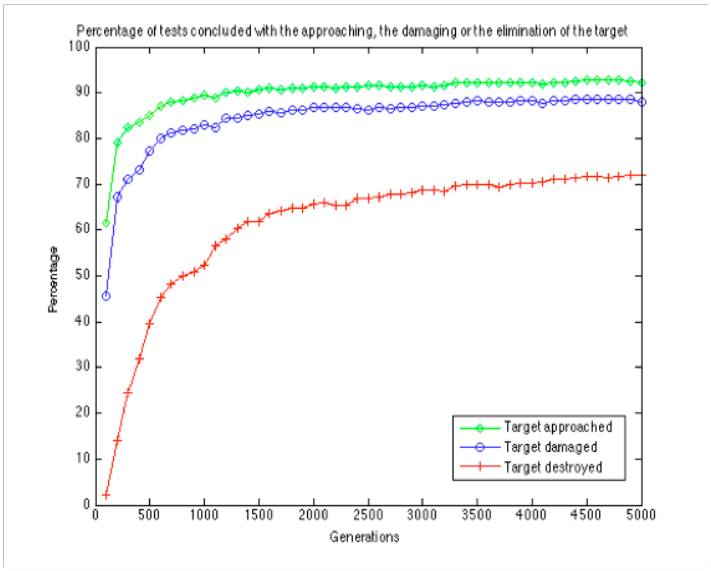


Figure 6 - Percentage of tests respectively concluded with the neutralization, the damaging and the simple approaching of the target when it is unable to move

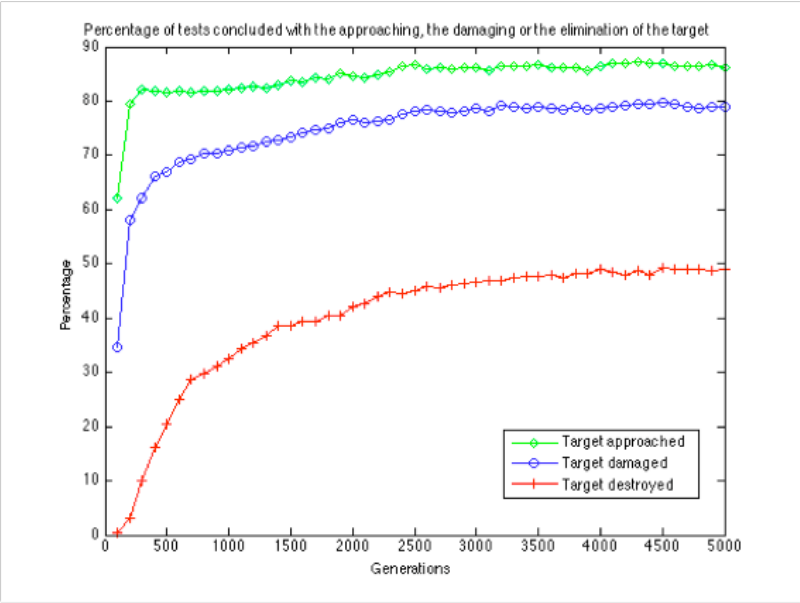


Figure 7 - Percentage of tests respectively concluded with the neutralization, the damaging and the simple approaching of the target when it is able to move, trying to escape from the approaching MAVs

Figures 6 and 7 show the results obtained with this experimental scenario, respectively with a fixed and a movable target. The simulations carried out using a fixed target produced a surprisingly good performance. On average, for the individuals belonging to the last generation, more than 70% of tests are successful, while 90% finishes with the target hit at least once. An example of the evolved behavior can be observed on Figure 8. It is important to highlight that in these simulations the MAVs were able to detect the presence of the environment boundaries, and some of them have clearly evolved a capability to exploit this strategy. They basically keep flying always close to an obstacle (which could be both a building, or one of the environment boundaries, until they get close to the target. Then they change their route trying to approach the target more accurately.

The performance of the swarms dramatically decreases when the target is moving, hence suggesting the need for the introduction of a form of communication within the MAVs that would positively affect the likelihood of successfully perform the task. In this experimental setup, only 50% of tests ends with the neutralization of the target, even if the percentages of tests concluded both with the approaching and with the damaging of the target are comparable with the ones obtained in case of a non-movable target.

New Experiment E: Optimization of the genetic algorithm

In order to improve the convergence speed of the evolutionary algorithm and to explore the solution space in a more efficient way, a new experiment has been carried out implementing three genetic operators different than before:

• the selection operator. As before, the best swarm of every generation is copied to the following one without any modifications (elitism), but then 94 pairs of parents are chosen for reproduction via a fitness-proportionate selection implemented through a "roulette wheel" sampling[4] [6];

• the crossover operator, which has been introduced in the form described by Montana and Davis [7]. Each of the selected pairs of parents generates a single child. In this way 94 new children are created. The crossover works in the following way: for each non-input neuron of the offspring, one

[4] In order to calculate the areas of the roulette's slices, the expected value for each individual has been measured as the ratio between its fitness and the average fitness of the entire population.

of the two parent is selected randomly and the child then inherits from the chosen parent the input connection weights to that neuron and the neuron's bias as well;

• the mutation operator, which affects all the 94 offsprings generated through crossover. For each neural network, 3 non-input neurons are selected randomly. The biases and all the incoming connection weights of the selected neurons are then subjected to a random mutation, adding to them a random value ranging between -0.5 and +0.5.

The remaining 5 individuals are created with randomly assigned connection weights and biases, in order to preserve the algorithm from the risk of a premature convergence.

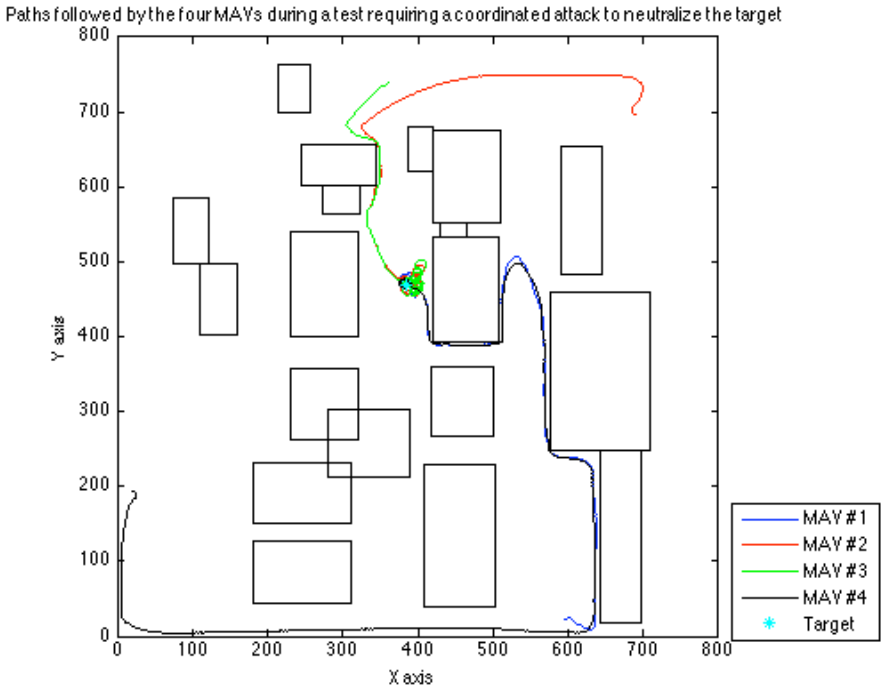


Figure 8 - Example of the flying paths followed by the MAVs when they have to neutralize a target unable to move, performing a cooperative action

The results obtained by this simulation, detailed on the second row of Table 2, look really poor if compared with those from experiment D[5]. The situation slightly improves if we scale the fitness values used to calculate the roulette's slices through the "sigma-scaling" method [6]. The results obtained (third row of Table 2) remain anyway worse than the ones obtained in experiment D.

Simulation	Av. fitness	Max fitness	Av. success %
D - non movable target	1023.8	1285.8	71.82
E - not scaled values	771.3	1133.7	47.47
E - scaled values	856.59	1253.9	56.69

Table 2 - Summary of the results obtained from the simulations of experiment E, and comparison with outcome of experiment D (with a target unable to move)

[5] Please consider that in this experiment the target is not able to move.

Some explorative analyses have also been conducted using a binary genome, instead of that with real values. Employing both Boolean and Gray Code encodings, with single and multi-points crossovers and different mutation rates, the results indicate a significant limit for the network to reach a weight set appropriate for the task, and therefore these conditions have been ignored.

Additional experiment (MSc Project): Generalization of the basic model to new environments

Thanks to the collaboration of two MSc students at the University of Plymouth, Franck Zetule and Francois Gautier, we have had the chance to carry out some additional tests while working at the same time on the experimental setups outlined in the previous sections. The two students were supervised by A. cangelosi and F. Ruini.

Zetule [8] replicated both experiments A and B, using a different simulated environment than the original one. His scenario represents "La Defense" area in Paris (Figure 10). Compared to the environment that we have previously seen on Figure 1, the one elaborated by Zetule is characterized by a smaller size (600 x 600 pixels vs 710 x 760 pixels) and fewer obstacles (tall buildings) present.

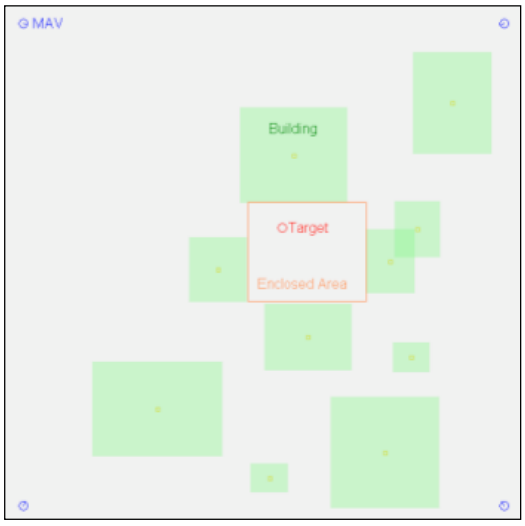


Figure 10 - The simulated environment used by Zetule [8] in order to replicate experiment B

Keeping the same settings as described in [3], the evolutionary process has demonstrated to run smoothly for experiment A. Figure 11 reports some of the results from Zetule's experiments.

In order to replicate experiment B, Zetule has then applied some minor modifications to the original fitness formula with the purpose of coping with the smaller environment (which in turns provoke a change in the balancing of the function, since one of its parameters is measured in pixels).

The fitness formula that Zetule used can be formalized as follows:

$$fitness = \frac{\nu}{8} - \alpha + (\frac{\beta}{10}) + (\sigma * 50) + (\phi * 10)$$

It differs from the original one for the smaller denominator applied to parameter β (10 instead than 50) and particularly for the introduction of the ν parameter, which represents the average difference between the distance of the MAVs from the target at the beginning and at the end of a test. The outcome of the experiment B replica is resumed in Figure 12.

Gautier [9], in another MSc project, was able to easily replicate experiment A on an environment with different shape and size, the National Library building complex in Paris (786 x 545 pixels). Gautier introduced a change to the original fitness formula, in his case multiplying parameter α per 1.5 in order to amplify its relative weight (operation which is compulsory given the fact that the simulated environment used is smaller than the one for which the fitness formula was originally optimized). The results of his simulation can be seen in Figure 13.

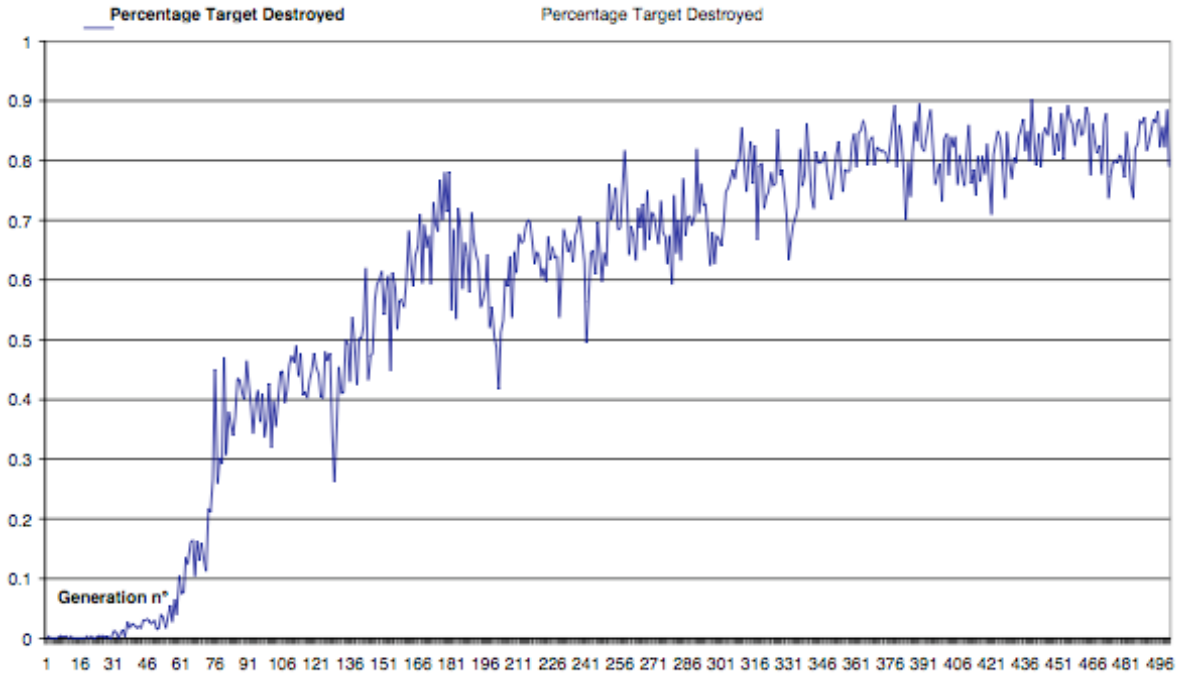


Figure 11 - Percentage of tests concluded successfully for the experiment A replicated by Zetule [8]

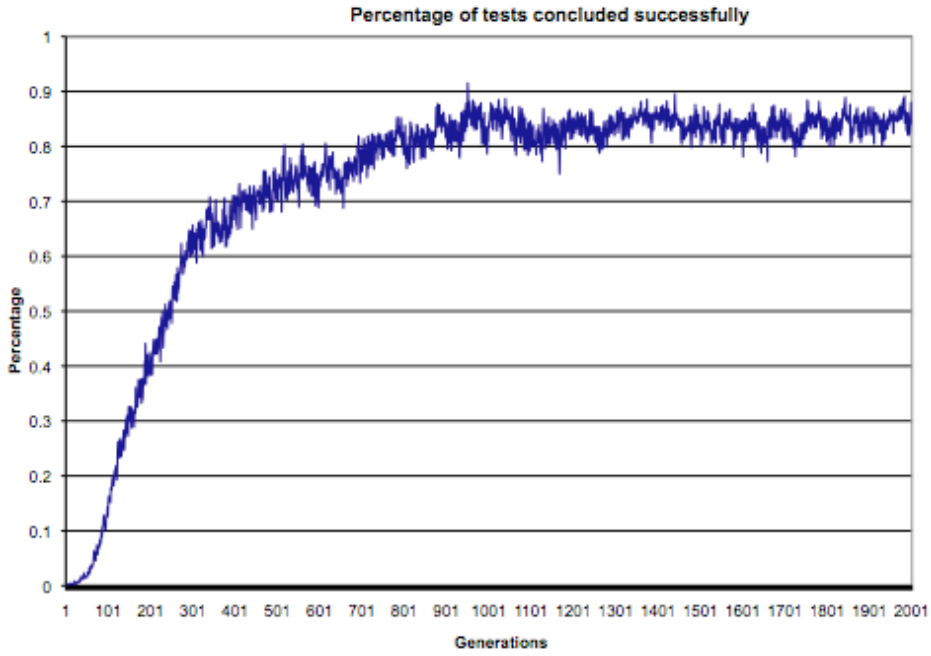


Figure 12 - Percentage of tests concluded successfully for the experiment B replicated by Zetule [8]

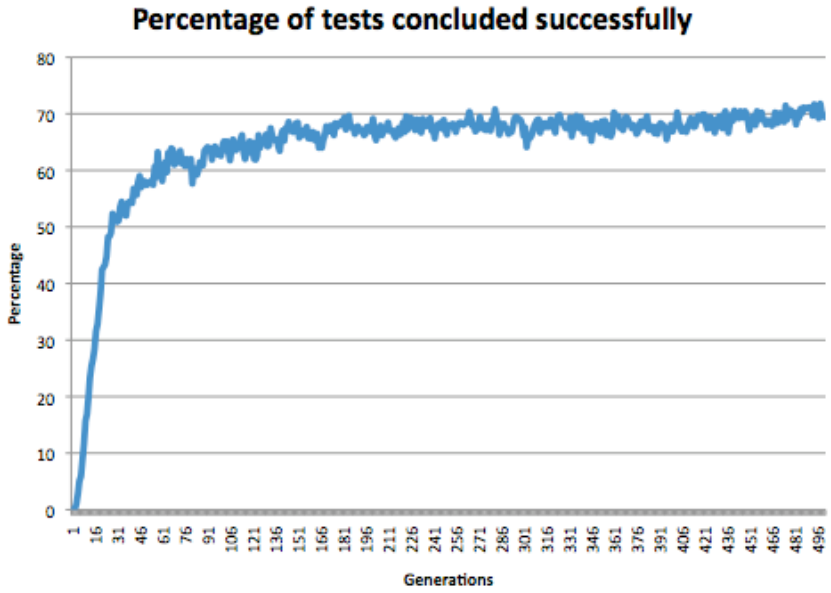


Figure 13 - Percentage of tests concluded successfully for the experiment A replicated by Gautier [9]

These MSc projects demonstrate that it is feasible to adapt the basic fitness formula described at the beginning of this report to any kinds of environment, with the only requirement of varying the α parameter depending on the environment size. Based on the data available, the value of this parameter might be calculated as the ratio between the area of the original environment and the one of the new environment[6].

Additional experiment (MSc Project): Multi-threading comparison

Starting from the assumption that most of the modern computers are usually endowed with more than one processor (and sometimes few cores for each processor), Gautier [9] also investigated the potential performances improvement offered by the application of multi-threading programming methodologies to our basic model.

Specifically, he has contrasted the performances scored by three different algorithms which can be briefly illustrated as follows:

• single-thread: all the swarms belonging to a certain generation are tested sequentially, using a single thread;
• multi-threads: within each generation, an amount of threads equal to the number of swarms which constitute the population is created. The various threads are assigned to the different processors - whenever they become available - and then elaborated;
• adaptive-threads: an amount of threads equal to the number of processors available on the machine is created at the beginning of the simulation. The swarms belonging to a certain generation are then associated to the different threads that will be immediately put on execution.

The multi-threading approaches have proven to outperform the single-threading methods, as expected. An interesting finding is that the use of different operating systems leads to different results. Particularly the experiments carried out on Microsoft Windows (both Vista and XP) have demonstrated that the so-called "adaptive-threads approach" generates better results than the "multi-threads approach". The opposite applies for the experiments carried out on Linux (Ubuntu). The results are resumed on Table 3.

[6] In this case, the value of the ratio is (760*710) / (786*545) = 1.26, which is a number close to the 1.5 used by Gautier and that has proven to be working.

Threading management algorithm / Operating system	Single thread	Multi-threads	Adaptive-threads
Windows XP	1	1.5	1.61
Windows Vista	1	1.37	1.53
Ubuntu Linux	1	1.68	1.37

Table 3 – Summary of the results obtained from the simulations carried out by Gautier [8] and aimed at investigating the advantages related to the adoption of multi-threading programming methodologies. The values reported are the ratios between the time required to carry out a simulation using a single-thread and the one required employing one of the others methods instead. Values greater than one represent a greater speed

Conclusions

Most funding is currently targeted at studying MAVs mainly from an ISTAR[7] perspective (see for example [10] and [11]). Our work focuses on the use for different kinds of tasks, in particular for providing MAVs with a strike capability in different scenarios. We can imagine at least two possible conditions in which MAVs could be employed as effective offensive instruments:

1) the first is related to counter-terrorism operations within urban environments. One of the most feared menaces by the Western countries' governments is a non-conventional attack coming from a terrorist group. As we have seen during the last years, particularly into the Middle East, the so-called "kamikaze strategy" is frequently employed, due to its simplicity both from an organizational and an economical perspective. One of the problems when facing menaces of this kind, is related to the fact that even if the attacker is identified it might become difficult to make him inoffensive. Using snipers would be possible, but could be difficult to deploy many of them along the strategical points of a crowded city, passing at the same time unnoticed (which is a compulsory requirement, since the attacker could suddenly change his strategy finding himself discovered). Needless to say, the "direct approach", involving the usage of a security task-force for approaching and neutralizing the target, is a highly risky operation. MAVs could be exploited as a valid alternative, or as an additional tool to existing approaches. Electrical propelled flying robots are in fact able to flight silently[8] and out from the typical line of sight of a person, allowing them to remain unnoticed while reaching their target. They would then be able to neutralize the attacker performing both a lethal (if equipped with a small amount of explosive able to generate a low-potential and very accurate detonation) or a non-lethal (using some chemical elements able to block the device starter, or employing devices like flashbang grenades in order to facilitate the intervention of a security task-force) action.

2) the second is related to an offensive operation into a warfare environment. Given their small size and portability, MAVs can be easily fit into soldiers' backpacks. This could allow special units, composed of just few soldiers, to carry with them a very flexible and powerful weapon. Once launched, each MAV (equipped as in the previous case with explosive devices) could in fact become part of a larger swarm and then cooperatively attack a target which would be instead impossible to offend by the soldiers' through their traditional weapons (e.g., gun, rifles, grenades,

[7] ISTAR: Intelligence, Surveillance, Target Acquisition and Reconnaissance.

[8] Of course the propellers produce a certain amount of noise, but it typically results impossible to be heard in a crowded place (and particularly if the MAV is flying at a sufficient height). Consider also that MAVs could easily switch their propellers while approaching the attacker from above.

etc.). The outcomes obtained by a MAV swarm acting in this way could be for some extents similar to the ones obtained through a low-potential missile. Despite the less damaging potential, the advantages are surely enormous. The first advantage is portability. It is simply not possible for small units to carry with them a "real" missile. Second advantage is flexibility. Units situated in different places could participate to the action just launching their MAVs at the same time, and then let them to join together and perform the attack against the target.

The research we have carried out up to date demonstrated how a MAV swarm could effectively navigate through unknown environments and reach a certain position into the space. The latest experiments elaborated have also shown that in principle these MAVs could be able to reach a certain level of coordination among them, in order to perform tasks requiring cooperation. Future experiments on the role of communication between MAVs will further investigate the role of coordination to improve performance in complex tasks such as those with moving targets.

Further discussions

Before concluding this report, we would like to spend few words about some interesting criticisms raised during the presentation of the ongoing works at some international conferences and workshops.

First of all, many people, particularly military personnel, expressed skepticism about the idea of armed aircraft flying absolutely autonomously on the skies of a western city. This is in fact a serious issue (not only due to the strictly policies applied by the air traffic control authorities) since the risk associated in letting potential dangerous robots flying without any control along a city where many civilians are present is high indeed. Even if the computer simulations used for the training of the robots would demonstrate a 100% accuracy of the controllers driving these aircraft, we would surely find more than one criticism coming from the people that would be considered responsible in the event of an accident. The problem relies for some extent on the methodology adopted, based on neural networks which basically are considered kinds of "black-boxes" on which no control is possible. In order to address this problem, anyway, it might be possible to remove from the MAVs the possibility of autonomously deciding when to carry out a detonation. This function could be "hardwired" instead, with a simple mechanism that automatically ignites the detonation when a certain set of conditions is non-ambiguously satisfied. This modification will make the MAVs not more dangerous than the manned military aircraft continuously flying over the skies of the entire world.

Another criticism frequently raised against this work is related to the lack of realism which affects the simulator we have developed. This issue has been highlighted several times, therefore it needs herein addressed. What we would like to do in our work is to demonstrate a principle, i.e. to demonstrate that neural networks could be successfully used as controllers for swarms of MAVs. The simulator we have developed serves exclusively this purpose. It does not aim to evolve neural network controllers immediately transferable to real aircraft. To some extents, we are assuming that the hardware platform we are simulating is able to perform the operations we ask it to do. For instance, when a MAV is on a certain position and we want it to move 50 centimeters forward along its facing direction, we assume the hardware is capable to guarantee the execution of this movement. As we will see in more details into the next paragraph, we are now working on a three-dimensional version of the simulator that can better capture some of these dynamics. Even if the implementation of such a modification could lead to a scenario which someone might consider more realistic than the previous one, our principal interest consists in looking for the possible evolution of new kinds of strategies - different than the ones emerged before, due to the more complex environment. From a technical point of view, moving from a 2D to a 3D simulator could be seen as the simple addition of a degree of freedom to the former model. Talking about realism,

we also need to take into account the fact that, differently from the most of experiments carried out on the evolutionary robotics field - which typically involve extremely simple wheeled robots (e.g., Khepera, e-puck, etc.) - the added value for reproducing a real physics environment in our work appears to be marginal. A certain degree of approximation, in fact, is always required when building a simulator (as it happens by definition for every kind of model). The point is how to find the correct balance of the trade-off between accurateness and simplicity of the simulator/model. This balance could be quite easy to identify when the simulated objects are wheeled robots, since the movement of a body on a plain surface is affected by few and well known forces (this is demonstrated by the enormous number of software able to correctly cope with this task, reviewed for example in [12]). The issues are different if we consider flight motion, where the amount of physics variables involved is extremely more complex than for wheeled/ground motion. As a first step, we have decided that the new 3D simulator will not be based on physics engines simulators, such as Open Dynamics Engine (ode.org), to keep the level of complexity manageable. Simulations will still guarantee that the MAV movement will be "realistic", even if not perfectly accurate from a physics point of view. This will allow us to focus on a first instance on the study of the coordination and communication strategies. Further extensions of this work with experiment on real MAVs will include the use of more realistic physics engine platforms.

Current and future developments

The next step of the project will be mainly focused on the development of a 3D simulator. This simulator will be used, first of all, to replicate all the results obtained so far in a simpler 2D scenario. Subsequently the 3D simulator will constitute the base platform on which the communication-related experiments will be carried out.

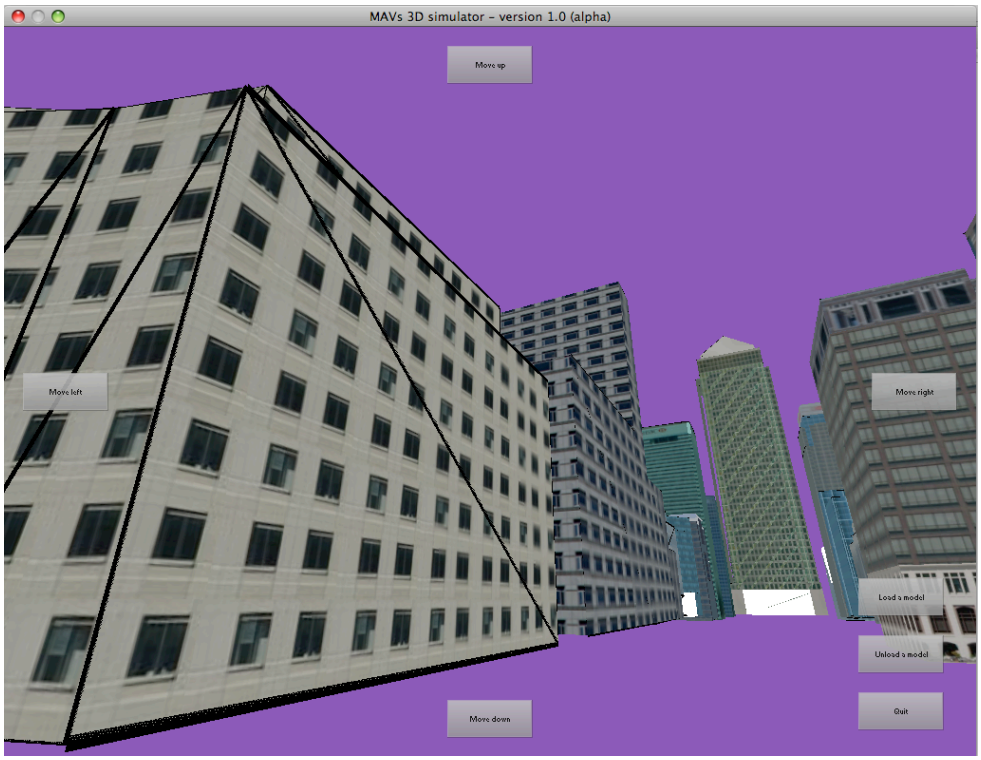


Figure 14 - Screenshot took from an alpha version of the new simulator. The environment which is possible to see is a 3D model of Canary Wharf, London, extrapolated from Google Earth and then re-elaborated through Blender

Amongst the various solutions evaluated for the design of the 3D MAV simulator, the choice about the 3D engine to employ has fallen on Irrlicht[9]. This is an open-source and multi-platform library -

[9] http://irrlicht.sourceforge.net/

easy to be included into any standard C++ application - which offers the best trade-off between easiness of use and richness of functions. In Figure 14 is possible to see a screenshot took from a very preliminary version of the simulator for the Canary Wharf district.

Given the added computational complexity of the 3D simulator, and following the introductory investigations on multi-threading programming by Gautier [9], we will focus also on the computational efficiency issue. New experiments will also benefit also from the P-ARTS: Plymouth Advanced Robot Training Suite project, a new simulation facility (based on the Apple's Xgrid technology) recently awarded to the University of Plymouth by Apple Inc, as part of the ARTS programme (Apple Research & Technology Support).

Essential bibliography

[1] Nolfi, S., and Floreano, D. (2000) "*Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*" (MIT Press)

[2] Richards, M.D., Whitley, D., and Beveridge, J.R. (2005) "Evolving *Cooperative Strategies for UAV Teams*" (in Proceedings of GECCO 2005: Genetic and Evolutionary Computation Conference, ACM Press)

[3] Ruini, F., and Cangelosi, A. (2008), "*Distributed Control in Multi-Agent Systems: a Preliminary Model of Autonomous MAV Swarms*" (in Proceedings of FUSION 2008: 11th International Conference on Information Fusion)

[4] Ruini, F., and Cangelosi, A. (2008) "*Evolutionary Algorithm based Neural Network Controllers: an Application to MAV Swarms*" (in Proceedings of WIVACE 2008: Italian Workshop on Artifical Life and Evolutionary Computation)

[5] Barlow, G.J., Oh, C.K., and Grant, E. (2004) "*Incremental Evolution of Autonomous Controllers for Unmanned Aerial Vehicles using Multi-objective Genetic Programming*" (in Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems)

[6] Mitchell, M. (1998) "*An Introduction to Genetic Algorithms*" (MIT Press)

[7] Montana, D.J., and Davis, L. (1989) "*Training Feedforward Neural Network Using Genetic Algorithms*" (in Proceedings of the Eleventh International Joint Conference on Artificial Intelligence)

[8] Zetule, F. (2008) "*Multi-Agent Systems in Search and Destroy Scenario Evolved Using Genetic Algorithsm: Research on Selection Algorithms*" (University of Plymouth, final project for the MSc on Interactive Intelligent Systems, supervised by Professor Angelo Cangelosi)

[9] Gautier, F. (2008) "*Evolving Neural Network using Genetic Algorithm in a Prey-Predator Scenario: Optimization by Threads*" (University of Plymouth, final project for the MSc on Interactive Intelligent Systems, supervised by Professor Angelo Cangelosi)

[10] House of Commons, Defence Committe (2008) "*The contribution of Unmanned Aerial Vehicles to ISTAR capability*" (Thirteenth Report of Session 2007-08, http://www.parliament.uk/defcom)

[11] Sullivan, J.M. (2006) "*Evolution or Revolution? The Rise of UAVs*" (IEEE Technology and Society Magazine, Vol. 25-3)

[12] Craighead, J., Murphy, R., Burke, J., and Goldiez, B. (2007) "*A Survey of Commercial & Open Source Unmanned Vehicle Simulator*" (in Proceedings of the 2007 IEEE International Conference on Robotics and Automation)

EOARD GRANT 073075

COMMUNICATION AND DISTRIBUTED CONTROL IN MULTI-AGENT SYSTEMS

Principal Investigator: Professor Angelo Cangelosi
PhD Student: Mr Fabio Ruini

Year 2, interim report at month 24

ABSTRACT

This report illustrates the latest developments in the context of our research, which is focused on the design of autonomous controllers for MAV teams. Work in year 2 focused on extending the previously developed 2D computer model to a more realistic 3D model of the UAVs and their flying environment. This new software simulator is being used for replicating and validating all the experiments carried out in the last months. In year 3 new and more complex experimental setups will be designed as well to complete the investigation of communication strategies in the UAV swarm.

DELIVERABLES (PUBLICATIONS)

Ruini F., Cangelosi A. (2009). Extending the evolutionary robotics approach to flying machines: an application to MAV teams. *Neural Networks*, 22, 812-821 (journal publication)

Ruini F., Cangelosi A., Zetule F. (2009). Individual and cooperative tasks performed by autonomous MAV teams driven by embodied neural networks controllers. *Proceedinsg of IJCNN-09 International Joint Conference on Neural Networks*, Atlanta, June 2009.

WORK IN PROGRESS ON THE 3D SIMULATOR

Moving to a 3D model implies providing the agents with two degrees of freedom (DOFs) more than what they had available in the 2D simulator (in the previous scenario, the MAVs were only allowed to rotate their body clockwise or counter-clockwise, essentially relying on a single DOF). If we consider an orthogonal reference system, built around the centre of mass of the simulated aircraft and moving with them, we can identify three class of rotations allowed to the MAVs: yaw, pitch, and roll. In more details: yaw is a rotation of the aircraft around his vertical axis; pitch and roll refers to the rotations around the wing-to-wing and the front-to-rear axis respectively (see Figure 1).

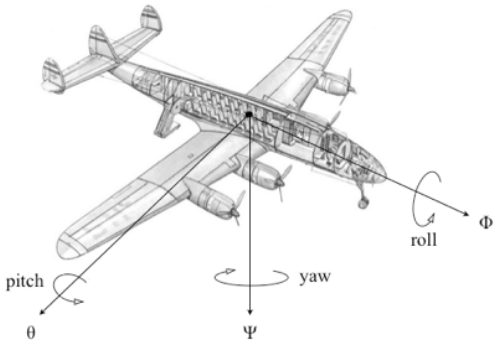


Figure 1 – The degrees of freedom (DOFs) typically associated to a fixed-wings aircraft

Each MAV's controller is implemented through a feed-forward neural network. The information the network receives in input is: (1) the distance from the target; (2) the horizontal angle between the target and the MAV, considered in relation to the heading of the latter; (3) the difference in altitude between the target and the aircraft; the current absolute MAV's (4) roll and (5) pitch angles. The output layer is made of continuous neurons incrementally modifying the MAV's rotation around the three axis. An additional Boolean output neuron implements the 'final action' the aircraft have to perform once reached the target.

The evolution toward neural networks exhibiting the desired behaviour is accomplished thorugh the use of a genetic algorithm. A starting population consisting in 100 controllers is created assigning to each one random synaptic weights and biases in the [-10.0; +10.0] range. Each controller is univocally assigned to an entire MAVs team. All the MAVs belonging to the same team are therefore driven by the same neural network. Teams are made of 4 MAVs each and are tested for 12 epochs, with the target deployed in as many different positions. The aircraft start a test epoch distributed next to the four environment corners, having a certain reserve of energy (which decreases over time, provoking the crash of the MAV in case it runs out). The reference environment is a 3D area, sized 1,500x,1000x600 Irrlicht's GUs[1], free of any obstructions. Once all the teams of the current generation have been tested for their ability to reach the target, their performances are evaluated according to a specific fitness function. The best 10 are then selected for reproduction. Each of them generates 9 copies of his genome, that are affected by random mutation: all the connection weights and biases have a 0.15 probability of being modified by meaning of random mutation in the [-0.2; +0.2] range. Elitism operator is also used in order to preserve the best controller in any given generation. 9 new controllers, with randomly assigned connection weights and biases, are created at each new generation. The entire process is iterated for 10,000 generations and repeated 20 times in order to limit as much as possible the effects due to randomness.

Many different neural network architectures have been tested (see the resume in Table I). Those are various feed-forward network's topologies, relying or not on a hidden layer, and receiving in input continuous or discretised values. Also

[1] Irrlicht is the name of the 3D engine used for writing the simulator. GU simply stands for Graphical Units.

architectures differ from each other according to how the roll is implemented (it can be absent, generated as a side effect of yaw, or independent) and for the presence of short-term memories, implemented as Elman or Jordan networks.

Table I. Neural network architectures used in the 3D simulator

NN arch.	Input	Hidden layer	Roll	Memory
1	D	No	No	No
2	C	No	No	No
3	D	No	With yaw	No
4	C	No	With yaw	No
5	D	No	Independent	No
6	C	No	Independent	No
7	D	Yes	With yaw	No
8	C	Yes	With yaw	No
9	D	Yes	Independent	No
10	C	Yes	Independent	No
11	D	No	With yaw	Jordan
12	C	No	With yaw	Jordan
13	D	Yes	With yaw	Elman
14	C	Yes	With yaw	Elman
15	D	No	Independent	Jordan
16	C	No	Independent	Jordan
17	D	Yes	Independent	Elman
18	C	Yes	Independent	Elman

PRELIMINARY RESULTS

The results obtained using the new simulator are resumed in Table II.

Table II. Results obtained using the 3D simulator

NN arch.	Average fitness	Maximum fitness	Success rate
1	126.5897	461.002	75.49
2	168.2282	454.7155	76.65
3	-130.3004	345.6638	43.63
4	-35.2221	407.9116	54.93
5	53.7617	406.5243	60.26
6	5.4618	391.213	57.49
7	-82.8986	360.323	47.54
8	49.8237	438.7521	63.96
9	119.3603	444.415	67.55
10	53.28	420.1875	60.28
11	-221.9481	232.8541	20.86
12	-160.9756	256.3188	24.72
13	-258.8055	113.8254	12.95
14	-265.2192	130.6172	15.5
15	-42.8824	311.3557	35.81
16	-18.4841	346.6911	45.3
17	-92.2134	291.5339	33.46
18	-85.5089	296.8809	34.21

As it was expected, architectures 1 and 2 are those that have scored the best absolute results. Anyway it is important to bear on mind that these two are just benchmark simulations. Since roll is not implemented, they tend to generate MAV's movements that are not 'realistic'[2]. One interesting aspect emerged is that the controllers seem working better when they can independently manage the roll, rather than when it is bound to yaw (notwithstanding directly operating this component increases the level of complexity of the neural network and the size of the solution space). Multi-layer networks have generally lead to better performances than those scored by networks where input and output layers are directly connected. The introduction of recurrences (memory), a part from having slowed down the evolutionary process, has not lead to an improvement in terms of performance. More ambiguous is the role played by sensorial input discretisation. In certain cases it has triggered better results than the use of continuous input; in other cases it has lead to a performance decrease.

MULTI-THREADING

The increased level of complexity involved in using a 3D simulator has stimulated an investigation on the effects generated by the employment of multi-threading programming procedures. The source code of the evolutionary engine (which has been disincorporated from the graphical part in order to reduce the computational load required by the evolution) has been modified accordingly to exploit all the computational power available on the computer system used. The multi-threading portion of the algorithm simply provides to distribute the various MAV teams to be tested on the cores available at any given moment.

The benefits coming from this modification, even if positive, have been lesser than our expectations. Switching from the use of 1 to 8 cores, we were expecting a corresponding code execution speed boost in terms of at least +600% (a value slightly smaller than the theoretical maximum +700%, which can not be reached because of the inevitable overheads). In reality the performed analysis have shown how the performance increment takes place only with a reduced magnitude. 10 experiments have been carried out, where 50 generations have been evolved using neural network architecture 9, using both single and multi-threading. In the first case, completing the evolution has required 150,354.8 msec on average; 27.765.1 msec in the latter. The velocity increment obtained is surely significant (+441.52%), but far from the theoretical +700% obtainable.

CONCLUSIONS

In this report we have analysed the recent work that have been done in order to extend the pre-developed 2D simulator in a new 3D model. The experience of the authors in testing multi-threading programming methodologies has been described as well.

[2] At the actual stage, the 3D simulator does not rely on any physics engine. Nonetheless the desired outcome is to make the MAVs exhibiting movements that are not in contrast with the major laws of physics.

The realism level of the previous model has been incremented. The modifications made constitute a further step toward the possible transfer of controllers developed in simulations to real robotics platforms. Despite a general decrease of the MAVs performances in carrying out tasks that were easily performed in the previous 2D simulator, the evolutionary process has lead anyway to positive results for the new model also. The performance decrease was an expected result, since adding new degrees of freedom to the simulator significantly augments the size of the solutions space that the evolutionary algorithm has to explore while looking for a proper configuration of synaptic weights and biases.

The results coming from the investigation about multi-threading are not solid and generalisable enough in order to extract definitive conclusions from them. They can be considered findings, suggesting that switching to parallel programming methodologies must be carefully planned, since it does not necessarily lead to the expected improvements in terms of simulations' execution speed. In the specific case, given the relative quickness in evaluating a single MAV team, a possible solution for optimizing the execution speed of the code while using multi-threading methodologies might be redistribute on any available core a larger number of teams. In this way, the impact of the overhead should be reduced. A part from replicating the experiments carried out on the 2D simulator and introducing new more interactive experimental setups, future work will also focus on clarify results.

Individual and Cooperative Tasks performed by Autonomous MAV Teams driven by Embodied Neural Network Controllers

Fabio Ruini, Angelo Cangelosi, Franck Zetule

Abstract— **The work presented here focuses on the use of embodied neural network controllers for MAV (Micro-unmanned Aerial Vehicles) teams. The computer model we have built aims to demonstrate how autonomous controllers for groups of flying robots can be successfully developed through simulations based on multi-agent systems and evolutionary robotics methodologies. We first introduce the field of autonomous flying robots, reviewing the most relevant contributes on this research field and highlighting the elements of novelty contained in our approach. We then describe the simulation model we have elaborated and the results obtained in different experimental scenarios. In all experiments, MAV teams made by four agents have to navigate autonomously through an unknown environment, reach a certain target and finally neutralize it through a self-detonation. The different setups comprise an environment with various obstacles (skyscrapers) and a fixed target, one with a moving target, and one where the target (fixed or moving) needs to be attacked cooperatively in order to be neutralized. The results obtained show how the evolved controllers are able to perform the various tasks with an accuracy level between 72% and 94% when the target has to be approached individually. The performance slightly decreases only when the target is both able to move and can only be neutralized through a coordinated operation. The paper ends with a discussion on the possible applications of autonomous MAV teams to real life scenarios.**

I. Introduction and Related Work

During the last decade several studies have been carried out on both wheeled and underwater autonomous vehicles driven by embodied neural network controllers (e.g. [1] and [2]). At the same time, the application of same principles to flying robots has not yet been thoroughly investigated. With the only notable exception of the systems developed by Floreano [3], Holland [4] and Buskey [5] it seems that current approaches on the development of autonomous controllers for aircraft mainly rely on techniques other than neural networks. Examples of these methodologies are behaviour-based robotics [6], genetic programming [7][8], evolution-based path planning [9], modeling field theory [10], and graph search methods [11].

In this study we use a multi-agent system (MAS) based on evolutionary robotics methodologies [12] to develop controller for MAVs for autonomous navigation, including obstacle-avoidance and target reaching, in unknown environments. We are interested in investigating how local interactions between many autonomous and independent MAVs could in turn lead to an observable (and therefore exploitable) higher level collective behaviour. Derived from complex systems sciences, our idea is that continuous low-level interactions between identical individuals, each of them owning just a minimal knowledge of the surrounding environment, could lead to the deployment of MAV teams where each aircraft acts independently from the others, still being able to take part in a bigger task. The combined use of evolutionary robotics and multi-agent systems will make possible to obtain collective behaviours without the need of designing a top-down cooperative strategy.

Distributed control, intended as the process of coordinating the movements of a number of agents in order to make them performing a collective task without using a central controller, is generally considered a notably interesting problem from both a technological and scientific perspective [1][13]. Good examples of the complexity involved in designing effective cooperative strategies for teams composed of many unmanned vehicles can be seen in the works made by Hussain [14] and Gaudiano [15]. In order to reduce this complexity, many studies regarding the behaviour of groups of Unmanned Aerial Vehicles (UAVs) have concentrated on flocking and swarming behaviour (e.g., [16] and [17]). We are not interested in replicating such a phenomenon. Instead, the approach we have chosen for studying the emergence of cooperation is based on the so-called "reactive strategies" [7].

The reactive strategy approach has several advantages with respect to those belonging to the other main category of "deliberative approach". Deliberative approach strategies focus on developing a specific flight path for each aircraft belonging to a team to follow (see for example [18]). Generating fixed routes in advance implies that a very good knowledge of the reference environment is available to the central controller (whether it is a human or a computer system). UAVs relying on such a kind of controller system could be therefore considered autonomous, in the sense that they will be able to autonomously follow a pre-planned flight path. But they would not have the ability of taking autonomous decisions, resulting therefore in a lack of intelligence (autonomy). This does not represent an issue for domains like civilian aviation, where all the needed information are immediately available. The lack of flexibility related to the "deliberative" approach becomes problematic instead if we try to apply the same principles to dynamic or unknown scenarios. These drawbacks have tried to been solved incorporating in deliberative approaches

Manuscript received December 15, 2008. Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-07-1-3075.

All authors are with the Adaptive Behaviour and Cognition Research Group, School of Computing, Communications and Electronics, University of Plymouth, Drake Circus, Plymouth, PL4 8AA, UK (phone: +44 (0)1752 586289; e-mail: fabio.ruini@plymouth.ac.uk).

some elements of adaptive replanning. The implementation of this kind of improvement requires equipping the aircraft with a set of sensors that makes them able to fetch previously unknown, non-accessible and/or non-existent information from the environment. The main idea in adaptive replanning is that a centralized controller generates a specific flight path for each UAV to follow based on the currently available information. UAVs strictly follow those paths until they detect some new elements through their sensors (e.g. an unknown enemy or an unexpected obstacles). When it does happens, the sensor information gathered is sent back to the controller, which may then decide to generate new flight paths for the entire team (or just part of it) and transmit them to the UAVs. A good example of adaptive replanning could be seen looking at the "UAV manager" concept elaborated by Rathinam et al. [19]. Despite the fact that adaptive replanning approach looks promising, many issues remain to be addressed in deliberative strategies. For example to decide when a replanning is required, and the amount of time needed to calculate and broadcast the new flight paths to the various UAVs are two non-trivial elements to consider. Scherer and colleagues [20] have recently identified a possible solution using two separated but interacting controllers that respectively act on a global and on a local level ("plan globally and react locally"). Even in this case, a good level of knowledge about the environment is still required.

Generally speaking, we might argue that it is the need for a central controller to be highly problematic. As highlighted for example by Wu and colleagues [21], distributed control is generally preferable since its non-critical reliance on any specific element can in turn guarantee increased reliability, safety and speed of response to the entire system. In addition to this we believe that a distributed control system has as well a better potential to produce adaptive and flexible solutions for the tasks we are interested in studying.

The main difference between the approach followed by us and a standard reactive strategy methodology as described in [7] mainly consists in the employment of a neural network controller instead of a properly defined decision tree. In both cases the controllers are subjected to an evolutionary process and therefore the use of computer simulators for the training phase results compulsory (unless we take into account some unusual alternatives, like a cable-array robot [22]).

The basic principle followed by us is to some extents similar to the ones proposed in [4] and [5] for the autonomous control of unmanned helicopters. The controller we use is an embodied neural network which outputs affect the aircraft's spatial orientation and its moving direction consequently. However our approach introduces at least three elements of novelty. The first is that we are focused on replicating the simplified dynamics of airplane-like UAVs instead than helicopters. Even employing a streamlined model as the one described herein, when compared to aircraft helicopters result much more flexible in adjusting their movements during the flight. If for example an unexpected obstacle arises, a helicopter could easily hover overhead, perform a 180 degrees yaw and then look for a different path to follow. When it comes to aircraft this kind of behaviour is not possible, so the on-line adjustments to the current route need to be extremely accurate. The only work to our knowledge where neural networks are applied to the control of non-helicopters or blimps aerial vehicles is the one made by Floreano and colleagues [3]. Furthermore, another major novelty consists in our decision of implementing a basic obstacle-avoidance mechanism, which represents an additional challenge to be addressed by the controller. Traditionally, obstacle-avoidance behaviour has not been taken into account in studies regarding UAV path planning (problem that affects also Floreano's investigation). As pointed out by Rathbun [9] this is mainly due to the fact that UAVs have usually been restricted to operate in areas that do not contain any other vehicles outside the control of the authority in charge of it. Rathbun's work - where an evolution-based path-planner results able to deal with movable and non-accurately estimated obstacles - constitutes one of the few meaningful exceptions to this trend. Finally, the controller we use is made of a single feed-forward neural network and not of different modules joined together, each of these dedicated to manage different sub-tasks as in [4] and [5]. The entire controller acts therefore as a single entity.

II. Description of the Model

As introduced in the previous paragraph, our approach requires the employment of a computer simulator for the evolution of UAVs' autonomous controllers. With the preliminary experiments carried out we have outlined the general specifics for a simulator of this kind. We have at the same time identified the minimal sensors requirements for allowing UAVs to perform navigation and search tasks both inside plain and obstacle-full environments (see [23]).

The structure of the simulator is quite simple. A team is composed by four MAVs[1], each endowed with its own neural network controller, identical to the ones of its teammates. At the beginning of a test, an "enemy" target is deployed somewhere inside the environment. The simulated scenario consists of a 2-D representation of Canary Wharf financial district in London. Starting from the four area's corners and facing the center of the environment, the MAVs have to fly toward the target attempting to eliminate it. In order to neutralize the target, one of the MAVs needs to perform a self-detonation when it is close enough to it (2.2 meters or less). A test ends when the target has been destroyed or no MAVs are still living. A MAV will die if it performs a detonation, if it attempts to exit from the environment's boundaries, if it collides against a teammate, if it runs out of energy, or if it crashes against a building.

Automatic target acquisition (ATR) is not provided by the MAVs. In this way they do not need to execute such an intensive computational task (even if the job could be effectively tackled cooperatively, as demonstrated for example by Dasgupta [24]). Our hypothesis consists on the presence of a satellite system that constantly monitors the target and broadcasts real-time information about its position to all the team members. In this way the MAVs - equipped

[1] Even if a proper classification is still lacking, a MAV can be roughly defined as a small-size UAV

with a GPS receiver - could easily calculate their distance from the target matching the two data sources gathered. Then a simple compass can as well easily allow the MAVs to determine the relative direction on which the target is. In our simulator each MAV is in fact fed with information about the distance between itself and the target, as well as the angle that separates the two agents based on the current MAV's heading. This information is received by the neural network by means of four input neurons: one encoding the distance (using discrete values), the others three the angle (using a Boolean representation of eight possible sub-spaces). The MAVs are also endowed with three ultra-sonic sensors, capable to detect the presence of an obstacle, which could be the target, a teammate or a building. This information is encoded using three continuous neurons, each of them activated with a value representing the distance from the current sensor and the closest obstacle perceived by it (if any within a certain range). The input neurons are connected to the neural network's hidden layer, made of 15 continuous neurons. The neural network's output layer consists of just two neurons. One output unit controls the MAV's steering direction (+/-20 degrees in the time unit); the other one is a Boolean neuron that, when it turns to 1, causes the MAV to carry out the detonation.

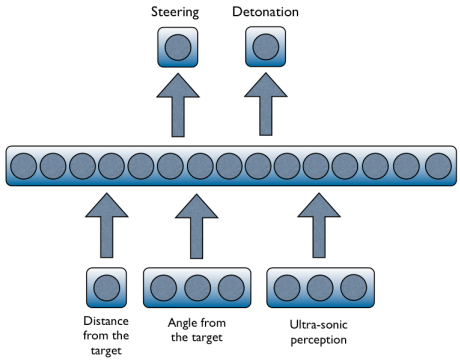


Fig. 1. Graphical representation of the neural network controller employed.

The fact that we are simulating an airplane-like motion implies the constraint, for the MAVs, of being always on movement. The speed is instead assumed as constant.

The evolution toward a controller able to perform the desired task is made possible by a genetic algorithm. An initial population of 100 teams is created with randomly assigned connection weights and biases ranging from -1.0 and +1.0. Each MAVs team is tested four times with the target deployed in randomly chosen positions (twice the target will be inside an "enclosed area" at the center of the environment, surrounded by buildings and with narrow entrances, twice it will be put outside this area). At the end of each generation the 20 individuals that has performed the best scores according to the fitness formula are selected for reproduction. Each team generates 5 copies of its genome, on which the mutation operator is then applied. Each gene of the copied genome is modified, with probability 0.25, of a random amount between -0.5 and +0.5. The only exception is for the best individual of the current generation, which generates a copy of its genome without any modifications (elitism). The resulting 100 individuals will constitute the new population at the next generation. The evolutionary process lasts for 2,500 generations and it is repeated 10 times with the results coming from all the different runs averaged in order to obtain more reliable data.

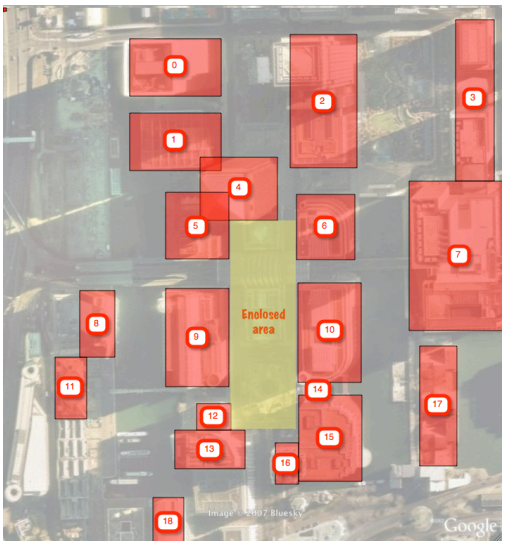


Fig. 2. The 2-D simulated environment used in our model. The obstacles, corresponding to the tallest buildings present in Canary Wharf area, have been highlighted.

The results coming from our preliminary analysis [23] show how the elaborated set up could lead quite easily to the proper evolution of the desired behaviour. At the end of the evolution, on average we have the 93.46% of tests successfully concluded into plain environments and 87.18% when obstacles are present.

III. Experiments

A. Movable target

In this experimental scenario, the target is able to detect a MAV approaching it. This new property of the target has been introduced to increase the complexity of the task and test the robustness of the model. During each time step, if a MAV is closer than 15 meters to the target, the latter can detect the aircraft with probability 0.5. In the event of detection, the target will then move away from the aircraft in order to maximize the distance from it. The target will remain in "MAV detected mode", and will keep moving away during every step, until the aircraft will die or the distance between the two agents will be over the 38 meters threshold.

The fitness formula used is the following:

$$f = -\alpha + (\frac{\beta}{50}) + (\sigma * 50) + (\phi * 10) \qquad (1)$$

where: α is the average distance (in pixels) between the target and the team member detonated closest to it, calculated basing on the various tests; β is the average amount of energy retained by the MAV detonated closest to

the target[2]; σ is the number of tests concluded by the given team with the elimination of the target; Φ is the total number of MAVs remained alive after the four tests (maximum 12). It is interesting to consider how the fitness formula we have decided to use does not require taking into account any information about the environment[3], like waypoints disseminated in specific places (as did for example in [4] and [20]). Navigation and obstacle-avoidance abilities emerge run-time as sub-tasks necessary for the completion of the main task, which is to neutralize the target.

Five simulations have been carried out where we vary the escaping speed of the target. Given the MAVs' flying speed of 55 km/h, the target speeds in the various simulations respectively correspond to 27.5 (Simulation A1: target speed = MAV speed/2), 18.33 (Simulation A2: target speed = MAV speed/3), 13.75 (Simulation A3: target speed = MAV speed/4), 11 (Simulation A4: target speed = MAV speed/5) and 9.16 (Simulation A5: target speed = MAV speed/6) km/h. The results obtained are summarized into Table I.

TABLE I
RESULTS FOR SIMULATIONS A

Sim	Av. fitness	Max fitness	Av. success %	Av. dist from the target (px)	Min. dist from the target (px)
A1	138.93	395.18	54.48	92.94	1.34
A2	198.08	409.67	78.28	107	1.09
A3	250.68	411.74	81.89	72.47	0.95
A4	258.72	409.9	83.3	70.03	0.98
A5	242.42	413.05	84.06	95.53	0.81

Observing the outcome of these simulations - particularly with regard to the average fitness - we can easily identify a kind of threshold. Simulations A3, A4 and A5 seem to perform equally well according to the various parameters measured. Simulation A2 produces a significantly worse performance for the average fitness, but could be considered performing reasonably well if we take into account both the maximum fitness and the average percentage of tests concluded successfully. In simulation A1 the success rate of the MAVs drops instead.

Comparing these results with the ones obtained using a static target, we can notice a general performance decrement. As clearly shown in Fig. 3, the difference is mainly concentrated on the average values, while the maximum ones (i.e., the best individuals/controllers within a certain generation) tend to reach similar level of performance. The main conclusion draw from this experiment is that the algorithm setup can evolve MAV controllers able to navigate through unknown environments and autonomously reach and destroy a target, not only when the latter is fixed on a certain position, but also if it is able to move away from them. The only constraint is that, in order to keep a reasonable success rate, the target should not be able to move faster than one third of the MAVs' speed. This is quite a reasonable assumption if we suppose that the target is not a vehicle, but a person instead. Considering that the moving speed of an average person moving in crowded environment could be approximated to 4-7 km/h while walking, and 15-20 km/h while running, we might argue that the evolved controllers are able to accomplish their task with a good degree of confidence even against a movable target[4].

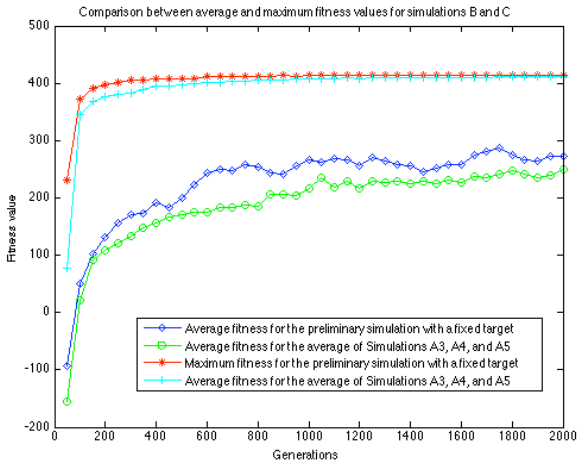


Fig. 3. Average and maximum fitness for simulations A3, A4 and A5 compared to the preliminary results obtained with a fixed target.

B. *Cooperative task*

The setup labeled as experiment B adds the constraint of requiring two MAVs to detonate against the target at the same time (i.e., within a limited maximum number of time-steps apart from each other, since the simulation works in discrete time steps) in order to neutralize it. The target begins each training epoch with the assigned status of "intact". When it happens that one of the MAVs detonates close enough to it (i.e., the same situation that in the previous setups would have provoked the elimination of the target), the target's status switches to "damaged". If a second MAV manages to detonate close enough to the target while this is still in the "damaged" mode, the target will be eliminated. Otherwise, after 10 time-steps of "damaged" state, the target will restore its original "intact" condition and the simulation will goes on as usual till the neutralization of the target or the failure of the MAV team.

In order to make the MAVs able to accomplish this task, we have provided them with the capability of gathering new pieces of information from the environment. Each member of the team is now able to detect both the status of the target ("intact" rather than "damaged") and the presence of a teammate within a certain distance. This information is given in input to the neural controller through two additional Boolean neurons. These two neurons implement a kind of logic OR. A part of being in the proximity of the target, in

[2] The MAVs start with 5,000 energy units. They spend 2.14 energy units per time, step, moving 2.24 meters far.

[3] For the sake of accuracy the size of the environment is used in order to scale some of the input values provided to the neural networks. Anyway, it has been proved that the neural network is able to evolve for carrying out the desired task even using non-scaled input values.

[4] Consider that a typical MAV platform, as could be the Aerovironment's WASP III, is able to reach a speed of 65 km/h (for full specifications look at: http://www.avinc.com/downloads/Wasp_III.pdf). One third of this speed roughly corresponds to 21.5 km/h, which is a value comparable to the 15-20 km/h suggested as the maximum speed reachable by an average person running.

order to decide the proper moment in which to detonate a MAV needs in fact to know that there is a teammate close to it, or that the target has recently been damaged (i.e., it is currently on the "damaged" status), or that both conditions (closeness and damaged) are true. Three neurons have been added to the hidden layer as well, in order to make the neural network able to cope with the increased amount of information collected from the environment.

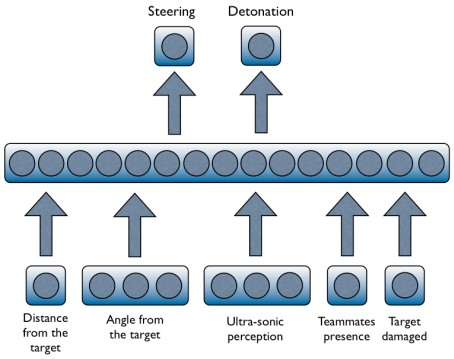


Fig. 4. The neural network architecture used for experiment B.

The fitness formula has been also modified in order to let the new desired behavior to evolve. We have now introduced the concepts of "target approached" and "target damaged". At the end of a test, we define the target as "approached" if at least one MAV has detonated within a 56 meters range from it. The target is considered "damaged" instead if at least one MAV has managed to hit it. These modifications tend to recreate what we could call an incremental evolutionary process (though if pursued in a different way than what has been done for example by Barlow and colleagues [8]). The MAVs initially learn how to perform the simplest sub-tasks (avoiding obstacles and approaching the target) and then progressively move toward the more complicated sub-tasks (damaging and neutralizing the target respectively), which in turns make the accomplishment of the overall task possible.

Putting all together, the new fitness formula is:

$$f = (\gamma * \frac{\omega}{4}) + (\eta * \frac{\omega}{2}) + (\lambda * \omega) + (\phi * 10) + (\frac{\beta}{50}) \quad (2)$$

where: Υ is the number of tests concluded with at least one MAV "approaching" the target; η is the number of tests concluded with at least one MAV "damaging" the target; λ is the number of tests concluded successfully and $\omega = 50$ (ω is just a parameter arbitrary chosen in order to assign different specific weights to Υ, η and λ). Parameters Φ and β have a similar meaning to the ones they have in (1), as they respectively represent the total number of MAVs survived at the end of the all tests and the average amount of energy retained by the MAV that had eventually neutralized the target. Consider that now every team is tested 12 times and the evolutionary process lasts for 5,000 generations.

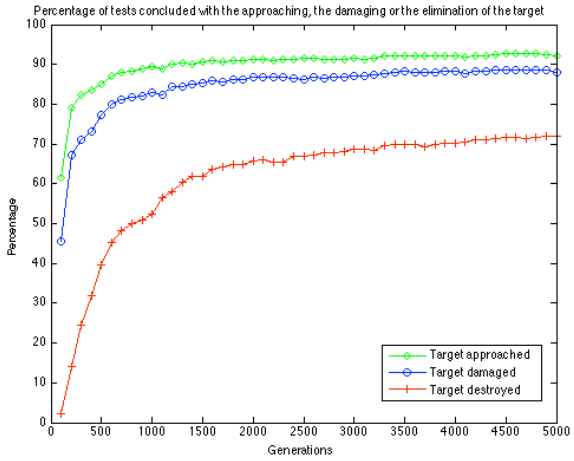


Fig. 5. Percentages of tests respectively concluded with the approaching, the damaging and the neutralization of the target when it is fixed and it has to be attacked cooperatively.

Fig. 5 and 6 show the results obtained with this experimental setup, respectively with a fixed and a movable target. The simulations carried out using a fixed target have produced a surprisingly good performance. On average, for the individuals belonging to the last generation, more than 70% of tests are successful, while 90% finishes with the target hit at least once. An example of the evolved behavior can be observed on Fig. 7.

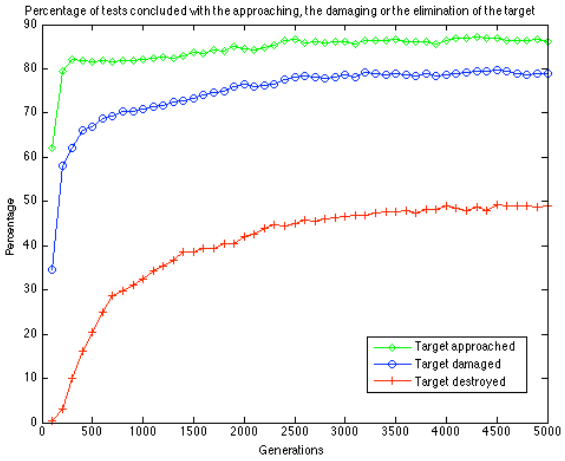


Fig. 6. Percentages of tests respectively concluded with the approaching, the damaging and the neutralization of the target when it is able to move and it has to be attacked cooperatively.

The performance of the teams dramatically decreases when the target is moving, hence suggesting the need for the introduction of a form of communication within the MAVs that would positively affect the likelihood of successfully complete the task. In this experimental setup, only 50% of tests ends with the neutralization of the target, even if the percentages of tests concluded both with the approaching and with the damaging of the target are comparable with the ones obtained in case of a non-movable target. Furthermore, we have to consider that we are illustrating average results referred to an entire population. It means that, inside this

population, the likelihood of having MAVs team particularly good in performing the desired task is extremely high.

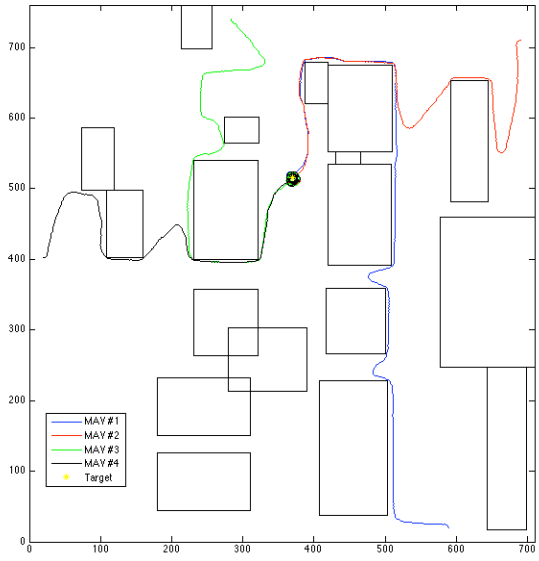


Fig. 7. Flight paths followed by the members of a team belonging to the last generation in order to reach the target and attack it cooperatively.

C. *Workarounds on the genetic algorithm*

In order to improve the convergence speed of the evolutionary algorithm and to explore the solution space in a more efficient way, a new experiment has been carried out implementing three genetic operators different than before:

1) Selection operator. As before, the best team of every generation is copied to the following one without any modifications, but then 94 pairs of parents are chosen for reproduction via a fitness-proportionate selection implemented as a "roulette wheel" sampling[5] [25].
2) Crossover operator, which has been introduced in the form described in [26]. Each of the selected pairs of parents generates a single offspring. In this way 94 new individuals are created. Crossover works in the following way: for each non-input neuron of the offspring, one of the two parents is selected randomly; the child inherits from the chosen parent the input connection weights to that neuron as well as the neuron's bias.
3) Mutation operator, which affects all the 94 offspring generated through crossover. For each neural network, 3 non-input neurons are selected randomly. The biases and all the incoming connection weights of the selected neurons are then subjected to a random mutation, adding to them a random value ranging between -0.5 and +0.5.

The remaining 5 individuals are created with randomly assigned connection weights and biases, in order to preserve the algorithm from the risk of premature convergence.

The results obtained by this new setup, detailed on the second row of Table 2, have highlighted a strong performance decreasing if compared with those coming from experiment B. The situation slightly improves if we scale the fitness values used to calculate the roulette's slices through the "sigma-scaling" method [25]. The results obtained in that (third row of Table II) remain anyway worse than the ones generated by experiment B.

[5] In order to calculate the areas of the roulette's slices, the expected value for each individual has been measured as the ratio between its fitness and the average fitness of the entire population.

TABLE II
COMPARISON BETWEEN SIMULATIONS B AND C

Sim	Av. fitness	Max fitness	Av. succ. %
B – non movable target	1023.8	1285.8	71.82
C – non scaled values	771.3	1133.7	47.47
C – scaled values	856.59	1253.9	56.69

Some explorative analyses have also been conducted using a binary genome, instead of that with real values. Employing both Boolean and Gray Code encodings, with single and multi-points crossovers and different mutation rates, the results indicate a significant difficulty for the network to reach a weight set appropriate for the task, and therefore these conditions have been ignored.

D. *Generalization of the model*

For the purpose of analyzing how the elaborated model could be generalized to different simulated environment, we have carried out few experiments varying the reference scenario. Measured in pixels, the original environment was sized 710x760. We have then created a new experimental setup - 600x600 large - with few obstacles present inside it.

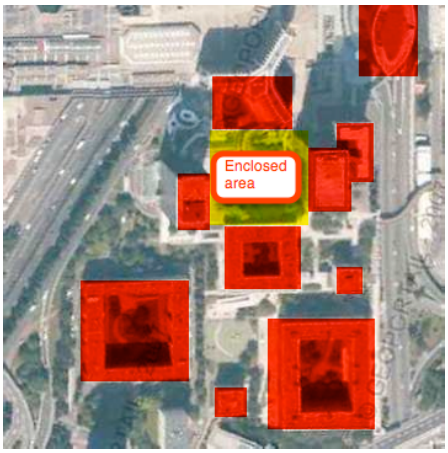


Fig. 8. The 2-D simulated environment used for the generalization experiments, with new obstacles layout (Paris, La Défense District).

The new environment (see Fig. 8) is smaller than the previous one and, summed to the presence of buildings and of a narrower enclosed area where the target is deployed, has provoked some troubles to the genetic algorithm in order to identify a proper set of connection weights and biases. In order to obtain a proper evolution, we have been required to modify the fitness formula in the following way:

$$f = \left(\frac{\upsilon}{8}\right) - (\alpha) + \left(\frac{\beta}{10}\right) + (\sigma * 50) + (\phi * 10) \qquad (3)$$

This formula differs from the original one for the smaller denominator applied to parameter β (10 instead than 50) and particularly for the introduction of the ν parameter, which represents the average difference between the distance of the

MAVs from the target at the beginning and at the end of a test. After 2,000 generations, the percentage of succeeded tests for this experimental setup has reached the 85% level.

Even if not conclusive, this further investigation has highlighted that it might be feasible to adapt the basic model described in this paper to any kinds of environments. It is not guaranteed that the original fitness formula could fit well to differently shaped and sized scenarios. The modifications made on this case have been marginal, but further studies are required in order to identify a general rule to follow when applying our model to different simulated environments.

IV. Discussion and Conclusion

Most researches are currently targeted at studying MAVs mainly from an ISTAR (Intelligence, Surveillance, Target Acquisition, Reconnaissance) perspective (see for example [27] and [28]). Our work focuses instead on the usage of MAVs for different kinds of tasks, requiring them having a strike capability available. We can imagine at least two possible scenarios in which MAVs provided with strike capability could be effectively employed.

The first scenario is related to counter-terrorism operations within urban environments. One of the most feared menaces by Western countries' governments is a non-conventional attack coming from a terrorist group. As we have seen during last years, particularly into the Middle East, the so-called "kamikaze strategy" is frequently employed, due to its effectiveness and simplicity both from an organizational and an economical perspective. One of the problems when facing menaces of this type is related to the fact that - even if the attacker is identified in advance - it might be difficult to make him inoffensive. Needless to say, the "direct approach", involving the usage of a security task force for approaching and neutralizing the target, is in fact a highly risky operation. MAVs could be exploited as a valid alternative to humans, or as an additional tool to existing approaches. Electrical propelled flying robots are in fact able to flight silently[6] and out from the typical line of sight of a person, allowing them to remain unnoticed while reaching their target. They would then be able to neutralize the attacker performing both a lethal (if equipped with a small amount of explosive) or a non-lethal (using some chemical elements able to block the device starter, or employing devices like flashbang grenades in order to facilitate the intervention of a land-based security task-force) action.

The second scenario is related to an offensive operation into a warfare environment. Given their small size and portability, MAVs can be easily fit into soldiers' backpacks. This could allow special units, composed of just few soldiers, to carry with them a very flexible and powerful weapon. Once launched, each MAV could in fact become part of a larger swarm and then cooperatively attack a target which would be instead impossible to offend by the soldiers through their traditional portable weapons. The outcomes obtained by a MAV team acting in this way could be just slightly minor than the ones obtainable through the employment of a low-potential missile. Despite the less damaging potential, the advantages, namely portability and flexibility, are surely enormous.

The research we have carried out up to date has demonstrated how a MAV team could effectively navigate through unknown environments and reach a certain position into the space. Particularly interesting is the fact that the neural networks employed in our simulations are very simple and they do not rely on any kind of short or long-term memories (thus confirming as well as extending the validity of what Buskey et al. have already found [29]). This would allow real MAVs to easily execute particular operations like the ones described above. The latest experiments elaborated have also shown that these MAVs could be able as well to reach a certain level of coordination among them, in order to perform tasks requiring cooperation. Future experiments will be mainly focused on the role played by explicit communication between MAVs. The purpose of introducing communication consists in investigating how its presence could lead to a better level of coordination between the agents and in turn allowing the teams to increase their effectiveness in performing complex tasks. The approach toward the evolution of a language will be based upon symbol grounding theory as introduced by Harnad [30] and then extended by Cangelosi et al. [31][32].

One potential criticism of the work we have done is related to the lack of realism which affects the simulator developed. Our aim was to demonstrate a principle through a computer simulation model, i.e. to demonstrate that neural networks could be successfully used as distributed controllers for teams of MAVs. The simulator we have developed serves primarily this purpose. It does not aim to evolve neural network controllers immediately transferable to real aircraft. To some extents, we are assuming that the hardware platform we are simulating is able to perform the operations we ask it to do. For instance, when a MAV is on a certain position and we want it to move 50 centimeters forward along its heading direction, we assume the hardware as capable to guarantee the execution of this movement. We are currently working on a 3-D version of the simulator that can better capture some of the real flight dynamics. Even if implementing such a modification could lead to a scenario with a higher degree of realism than the previous one, our principal interest consists in looking for the possible evolution of new kinds of strategies, different than the ones emerged before due to the more complex environment. From a technical point of view, moving from a 2-D to a 3-D simulator could be seen as the simple addition of a degree of freedom to the former model. A certain degree of approximation, in fact, is always required when building a simulator (as it happens by definition for every kind of model). The point is how to find the correct trade-off between accurateness and simplicity of the simulator/model. This balance could be quite easy to identify when the simulated objects are wheeled robots, since the movement of a body on a plain surface is affected by few and easily replicable forces (this is demonstrated by the enormous

[6] Of course the propellers produce a certain amount of noise, but it typically results impossible to be heard in a crowded place (and particularly if the MAV is flying at a sufficient height). Consider also that MAVs could easily switch their propellers off while approaching the attacker from above.

number of software applications able to correctly cope with this task [33]). The issues are different if we consider flight motion, where the amount of physics variables involved is extremely bigger than for wheeled/ground motion. As a first step, to keep the level of complexity manageable, we have decided that the new 3-D simulator will not be based on any physics engine. Simulations will still guarantee that the MAV movements will be "realistic", even if not perfectly accurate from a physics point of view. This will allow us to focus on a first instance on the study of coordination and communication strategies. Further extensions of this work with experiments on real MAVs will include the use of more realistic physics engine platforms.

Finally, we would like to consider the inclusion of other techniques for the MAV controllers. Modeling field theory [10][34] has been recently proposed as a learning technique for multi-agent simulation systems. One of the advantages of this approach is that of overcoming computational complexity and allowing better scaling up of the model capabilities, e.g. in terms of population size and internal representations. Future studies will explore the combination of modeling field theory within the agent control systems.

ACKNOWLEDGMENT

The authors would like to thank Francois Gautier for the work carried out related to some generalization experiments.

DISCLAIMER

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies and endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon.

REFERENCES

[1] G. Baldassarre, D. Parisi, and S. Nolfi, "Distributed coordination of simulated robots based on self-organization," in *Artificial Life*, Vol. 12(3), pp. 289-311, 2006.

[2] V. Kodogiannis, "Neuro-control of unmanned underwater vehicles," in *Int. Journal of Systems Science*, Vol. 37(3), pp. 149-162, 2006.

[3] S. Hauert, J.C. Zufferey, and D. Floreano, "Evolved swarming without positioning information: an application in aerial communication relay," in *Autonomous Robots* (in press).

[4] R. De Nardi, O. Holland, J. Woods, and A. Clark, "SwarMAV: a swarm of miniature aerial vehicles," in *Proc. 21st Int. UAV Systems Conf.*, 2006.

[5] G. Buskey, J. Roberts, P. Corke, P. Ridley, and G. Wyeth, "Sensing and control for a small-size helicopter," in *Experimental Robotics VIII*, Springer Berlin, pp. 476-486, 2003.

[6] M. Dong, and Z. Sun, "A behavior-based architecture for unmanned aerial vehicles," in *Proc. IEEE Int. Sym. on Intelligent Control*, 2004.

[7] M.D. Richards, D. Whitley, and J.R. Beveridge, "Evolving cooperative strategies for UAV teams," in *Proc. Genetic and Evolutionary Computation Conf. (GECCO 2005)*, 2005.

[8] G.J. Barlow, C.K. Oh, and E. Grant, "Incremental evolution of autonomous controllers for unmanned aerial vehicles using multi-objective genetic programming," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2005.

[9] D. Rathbun, S. Kragelund, A. Pongpunwattana, and B. Capozzi, "An evolution based path planning algorithm for autonomous motion of a UAV through uncertain environments," in *Proc. 21st Digital Avionics System Conf.*, 2002.

[10] R. Deming, L. Perlovsky, and R. Brockett, "Sensor fusion for swarms of unmanned aerial vehicles using modeling field theory," in *Proc. Int. Conf. on Integration of Intensive Multi-Agent Systems: Modeling, Evolution, and Engineering (KIMAS 2005)*, 2005.

[11] Y. Qu, Q. Pan, and J. Yan, "Flight path planning of UAV based on heuristically search and genetic algorithms," in *Proc. 31st IEEE Industrial Electronics Society Conf. (IECON 2005)*, 2005.

[12] S. Nolfi, and D. Floreano, *Evolutionary robotics*, MIT Press, 2000.

[13] G. Nitschke, "Emergence of cooperation: state of the art," in *Artificial Life*, Vol. 11(3), pp. 367-396, 2005.

[14] T. Hussain, D. Montana, and G. Vidaver, "Evolution-based deliberative planning for cooperative unmanned ground vehicles in a dynamic environment," in *Proc. Genetic and Evolutionary Computation Conf. (GECCO 2004)*, 2004.

[15] P. Gaudiano, E. Bonabeau, and B. Shargel, "Evolving behaviors for a swarm of unmanned air vehicles," in *Proc. IEEE Swarm Intelligence Symposium (SIS 2005)*, 2005.

[16] R.J. Bamberger Jr., D.P. Watson, D.H. Scheidt, and K.L. Moore, "Flight demonstrations of unmanned aerial vehicle swarming concepts," in *John Hopkins APL Tech. Digest*, 27(1), 41-55, 2006.

[17] J.J. Corner, and G.B. Lamont, "Parallel simulation of UAV swarm scenario," in *Proc. Winter Simulation Conference (WSC 2004)*, 2004.

[18] V. Ablavsky, D. Stouch, and M. Snorrason, "Search path optimization for UAVs using stochastic sampling with abstract pattern descriptors," in *Proc. AIAA Guidance Navigation and Control Conference*, 2003.

[19] S. Rathinam, M. Zennaro, T. Mak, and R. Sengupta, "An architecture for UAV team control," in *Proc. 5th IFAC symposium on intelligent autonomous vehicles (IAV 2004)*, 2004.

[20] S. Scherer, S. Singh, L. Chamberlain, and M. Elgersma, "Flying fast and low among obstacles: methodology and experiments," in *The Int. Journal of Robotics Research*, Vol. 27(5), pp. 549-574, 2008.

[21] A.S. Wu, A.C. Schultz, and A. Agah, "Evolving control for distributed micro air vehicles," *Proc. IEEE Conf. on Computational Intelligence in Robotics and Autonomation Engineers*, 1999.

[22] K. Usher, G. Winstanley, P. Corke, D. Stauffacher, and R. Carnie, "Air vehicle simulator: an application for a cable array robot," in *Proc. Int. Conf. on Robotics and Automation (ICRA 2005)*, 2005.

[23] F. Ruini, and A. Cangelosi, "Distributed control in multi-agent systems: a preliminary model of autonomous MAV swarms," in *Proc. 11th Int. Conf. on Information Fusion (FUSION 2008)*, 2008.

[24] P. Dasgupta, "A multiagent swarming system for distributed target recognition using unmanned aerial vehicles," in *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 38(3), pp. 549-563, 2008.

[25] M. Mitchell, *An introduction to genetic algorithms*, MIT Press, 1998.

[26] D. Montana, and L. Davis, "Training feedforward neural networks using genetic algorithms," in *Proc. 11th Int. Joint Conf. on Artificial Intelligence*, 1989.

[27] House of Commons, Defence Committee (2008), *The contribution of unmanned aerial vehicles to ISTAR capability*. Available: http://www.parliament.uk/defcom

[28] J.M. Sullivan, "Revolution or Evolution? The rise of the UAVs," in *IEEE Technology and Society Magazine*, Vol. 25(3), pp. 43-49, 2006.

[29] G. Buskey, G. Wyeth, and J. Roberts, "Autonomous helicopter hover using an artificial neural network," in *Proc. 2001 Int. Conf. on Robotics & Automation*, 2001.

[30] S. Harnad, "The Symbol Grounding problem," in *Physica D*, Vol. 42, pp. 335-346, 1990.

[31] A. Cangelosi, A. Smith, and K. Smith (editors), *The Evolution of Language*, World Scientific, 2006.

[32] A. Cangelosi, V. Tikhanoff, J.F. Fontanari, and E. Hourdakis, "Integrating language and cognition: A cognitive robotics approach," in *IEEE Computational Intelligence Mag.*, Vol. 2(3), pp. 65-70, 2007.

[33] J. Craighead, R. Murphy, J. Burke, and B. Goldiez, "A survey of commercial & open source unmanned vehicle simulators," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2007.

[34] L. Perlovsky, *Neural Networks and Intellect: Using Model-Based Concepts*, Oxford University Press, 2001.

Mr Fabio Ruini, Prof Angelo Cangelosi
University of Plymouth, UK
Centre for Robotics and Neural Systems

EOARD INTERIM REPORT for grant FA8655-07-1-3075

ABSTRACT

This report illustrates the latest developments in the context of our research, which is focused on the design of autonomous controllers for MAV teams. Recent work has been aimed to improve the new 3D computer simulator used for the evolutionary design of the aircraft's controllers. A new paradigm, namely incremental evolution, has been introduced with the purpose of smoothing the evolutionary process (and, together with the multi-threading introduced before, reducing the time required by simulations as well). The controllers are first evolved for a simple task (target reaching within a three-dimensional obstacle-free environment), then the same controllers are evolved further for more sophisticated scenarios (individual reaching of a non-cooperative target, and collective reaching of a stationary target). The results presented here come from ongoing experiments. Once these investigations will be completed, the project will move to its final stages, focused in creating a new experimental setup replicating an urban-like environment and testing/validating some of the controllers designed on simulations on a physical robotics platform.

NEW EXPERIMENTAL SETUPS

Since the previous report, the 3D computer model developed (see a Screenshot in Figure 1) has been improved from a technical point of view and experiments on various setups have been carried out. This has lead to a new publication (Ruini & Cangelosi, in press) accepted for the 2010 edition of the International Joint Conference on Neural Networks (IJCNN). A more updated article, including a detailed discussion of the latest results introduced herein, is currently in preparation and will be submitted to TAROS'2010 (11th Conference Towards Autonomous Robotic Systems).

On top of the basic test scenario (individual reaching of a stationary target; an example of this behaviour can be seen in Figure 2), the two new experimental setups elaborated respectively involve the reaching of a non-cooperative target (i.e., a target attempting to escape, at different speeds, from an approaching MAV), and the coordinated approaching of a stationary target (i.e., two MAVs have to coordinate among themselves and reach the target area at the same time).

In the first case, the controllers have easily proven to be capable of performing the desired task. As expected, their success rate inversely correlates with the moving speed of the target: the faster the target, the lesser the success rate. The results obtained in the 3D scenario are pretty similar to those previously extrapolated using the 2D model (Ruini et al. 2009) (Ruini & Cangelosi, 2009). For targets moving at one fifth, one fourth and one third of MAVs' speed, the best success rate obtained (regardless the neural architecture used for designing the controller) is about 90%. A target moving at half the MAVs' speed results instead much more difficult to be

approached and the success rate consequently drops into the 55.0-60.0% range. An example of non-cooperative target reaching behaviour is shown in Figure 3.

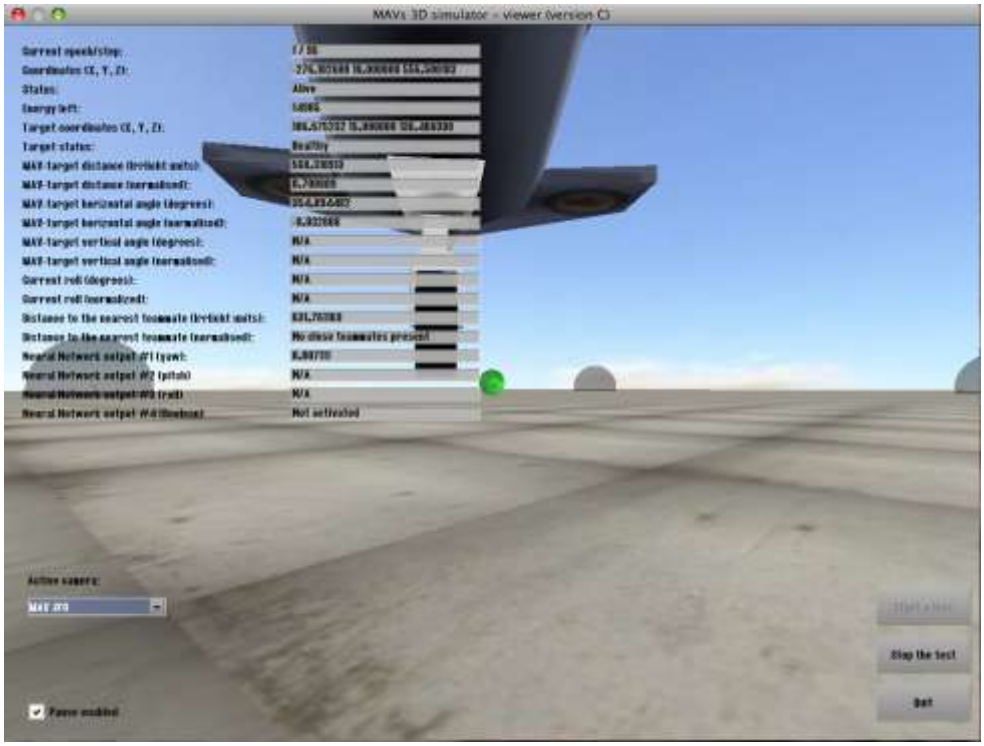


Figure 1 - Screenshot of the 3D simulator. View from the camera installed on one of the MAVs involved in the task

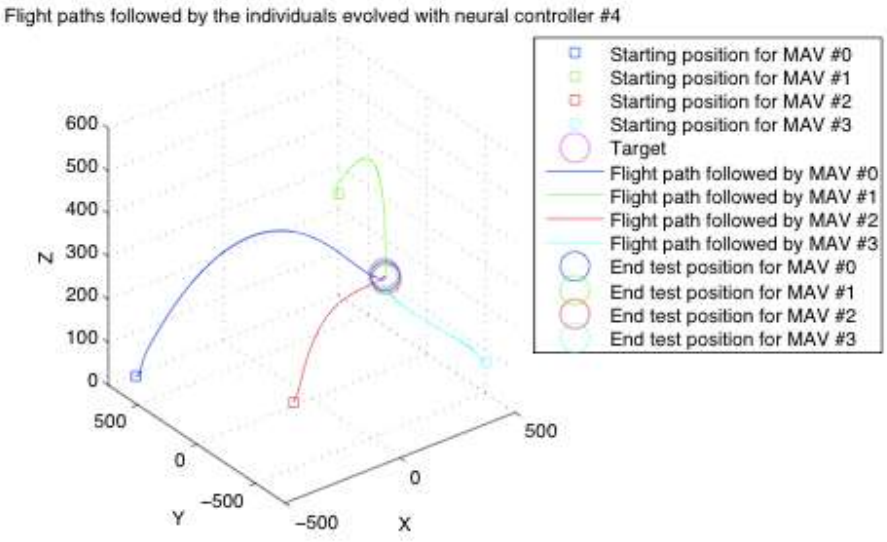


Figure 2 - Flight paths followed by a team of 4 MAVs sharing the same controller, within the basic scenario. Each MAV acts independently from the others

Regarding the setup focused on coordinated behaviour, the results generated by the evolved controllers have been generally worse than what we were expecting. Only in the simplest cases (MAVs uniquely allowed to yaw, thus flying always at the same altitude and parallel to the ground) the evolutionary process has lead to convincing and robust solutions. The evolved behaviour consists in the first MAV approaching the target and, rather than activating the Boolean output of his neural network

controller (operation that the MAVs have to perform within all the experimental setups in order to successfully perform the task they are involved in), engaging in a circular flying trajectory, waiting for at least a second MAV to arrive. When another teammate has arrived, both of them further approach the target in order to conclude the operation.

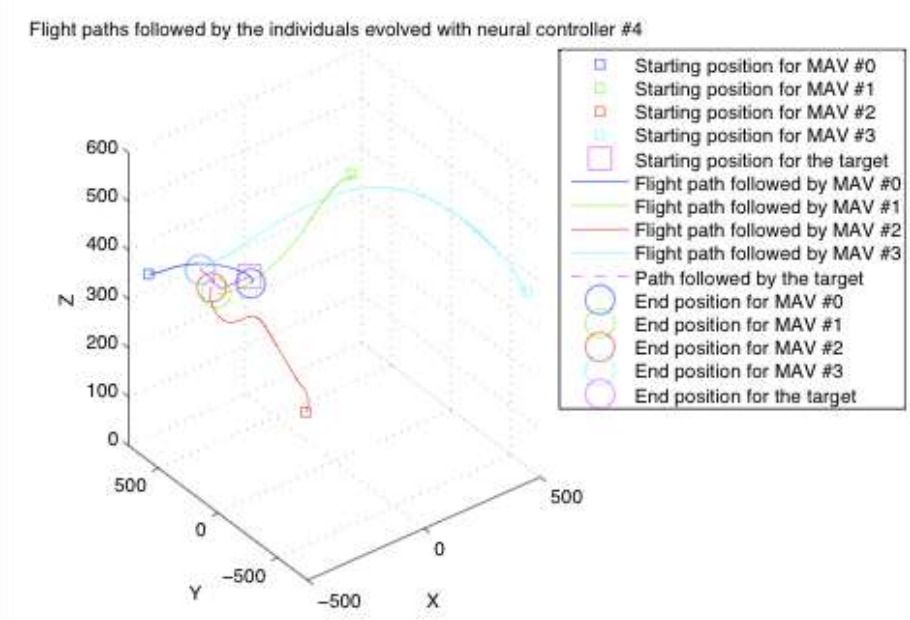


Figure 3 - Flight paths followed by a team of 4 MAVs sharing the same controller, within the second experimental setup (non-cooperative target reaching), for a target moving at one third of the MAVs' speed

This kind of behaviour is well exemplified by Figure 4, took from the 2D simulator.

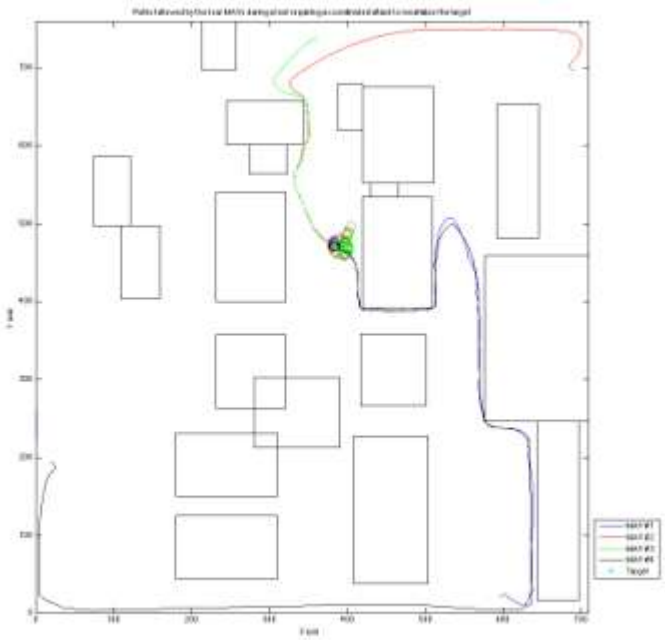


Figure 4 – Flight paths followed by a team of 4 MAVs sharing the same controller, within an experimental setup tested on the 2D simulator (coordinated target approaching and obstacle avoidance)

INCREMENTAL EVOLUTION

Incremental evolution (Petrovic, 1999) (Christensen & Dorigo, 2006) has proven to be a very successful methodology for improving the performances obtained by the evolutionary controllers. A basic experimental setup has been designed (as reported above) with a single MAV flying around an obstacle-free environment in order to reach a specific stationary target area. Then, the controllers evolved within this context have been subjected to a further evolutionary process - 5,000 generations long - where the MAVs were asked to perform the same task as before, this time against a non-cooperative (moving) target.

In setup B we previously identified a threshold in the target's speed. The performance of the controllers, if the target speed was exceeding that threshold (equal to half the MAV's speed), dropped significantly. Using incremental evolution this threshold has disappeared (or, at least, it is save to say that its effect has smoothened). This can be easily seen looking for example at the comparison between the results scored using controller architecture 11 (one of the most complex employed, since it involves a scenario where the MAVs are free to yaw, pitch, and roll) with and without incremental evolution, resumed in Table I.

Table I - Comparison between simulations with and without incremental evolution, for the experimental setup involving the reaching of a non-cooperative target moving at half the MAVs' speed

NN arch.	Av. fitness	Max. fitness	Av. succ. rate (%)	Max. succ. rate (%)
11 – non inc.	659.6893	1150.5	0.2243	0.5985
11 – inc.	554.3824	1167.4	0.3026	0.8108

The same methodology is currently being tested for the coordinated behaviour scenario. Since the neural network architecture used for this setup is different than those employed when the MAVs have to individually approach either a stationary or moving target, incremental evolution is implemented in a slightly more sophisticated way. In the coordinated behaviour scenario, the controllers rely on two additional input units. This affects, in turns, the number of connections present into the network (how strong this effect is depends both upon the presence of a hidden layer, and on the number of output units used). In our approach we simply add the new units and connections on top of an architecture evolved in the basic scenario (individual reaching of a stationary target) and set the connection weights to 0. Then we start a new evolutionary process, longer than the previous one (15,000 generations rather than 5,000), to allow the network to identify a new balance.

The preliminary results obtained so far are encouraging and seem supporting our hypothesis that this approach would prove successful to evolve cooperative behaviours within the 3D experimental setup. Further investigations will be carried out during the coming weeks in order to confirm this impression.

FUTURE WORK

The last part of the research will involve testing and validating on a real robotics platform the controllers designed in simulations. The work will be carried out - since May until September 2010 - at the Laboratory of Intelligent Systems (LIS) of the

École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, with the support of Prof Dario Floreano, Dr Jean-Christophe Zufferey, and Ms Sabine Hauert.

The robotic aircraft used will be the senseFly's *swinglet*[1]. Weighted about 400g and supporting up to 150g of additional payload, the swinglet is a flying-wing (80cm wingspan) MAV built out of expanded polypropylene (EPP), with an electric motor mounted at the back and two control surfaces serving as elevons (combining ailerons and elevator). In the configuration in use at the EPFL (Hauert et al., in press), the swinglet carries onboard a Toradex Colibri PXA270 CPU board[2] running Linux operating system. The board is connected to a Netgear WNDA3100 USB Wi-Fi dongle[3], which make the MAV capable of communicating over the standard 802.11n protocol, and to an autopilot built around a dsPic33 micro-controller[4] (Leven et al., 2009). In addition to providing an interface for receiving commands issued by the computer board, the autopilot takes care of controlling altitude, airspeed and turn rate of the MAV. This setup will allow us to work adopting the same methodologies already developed in the 2D simulator (Ruini et al., 2009) (Ruini & Cangelosi, 2009).

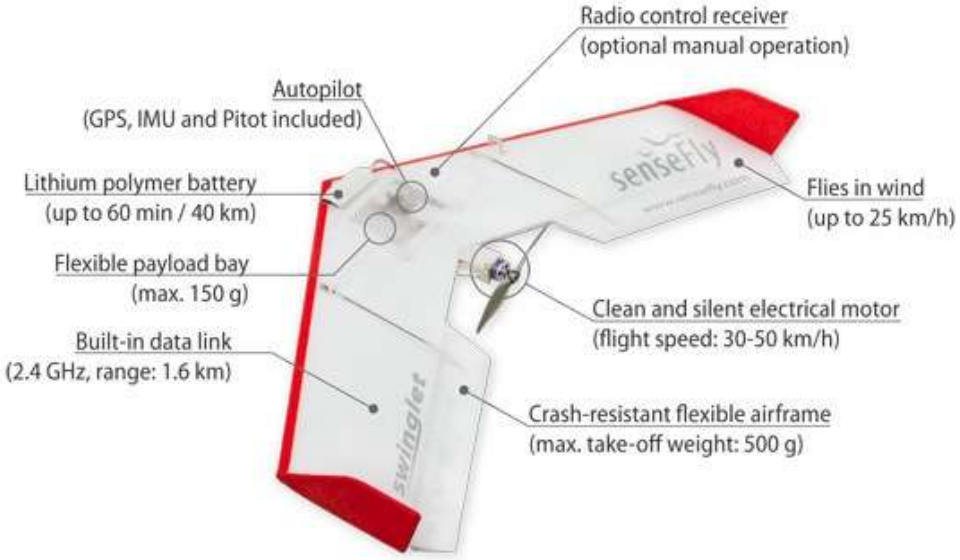


Figure 5 - senseFly's swinglet

Due to the impossibility, for technical reasons, of testing obstacle-avoidance behaviours using the test bed platform provided[5], the work will focus on two alternative topics: 1) formation flight, and 2) target reaching behaviour.

Formation flight is a well-known strategy (Lissaman & Shollenberger, 1970) (Weimerskirch et al., 2001) used by birds in order to save energy during movement. Recently, many attempts have been carried out to implement the same kind of behaviour within autonomous aircraft (NASA's AFF project[6]) (Naffin & Sukhatme,

[1] http://www.sensefly.com/products/swinglet/
[2] http://www.toradex.com/En/Products/Colibri_Modules/Colibri_PXA270
[3] http://www.netgear.com/Products/Adapters/RangeMaxWirelessNAdapters/WNDA3100.aspx
[4] http://www.microcontroller.com/news/microchip_dsPIC33.asp
[5] The minimum flight altitude provided by the onboard autopilot is about 50 meters. At that altitude is just impossible to deploy artificial obstacles the MAVs should attempt to avoid.
[6] http://www.nasa.gov/centers/dryden/history/pastprojects/AFF/index.html

2002) or spaceships (Aung et al., 2003) (Tillerson et al., 2003). The approach we will be following aims to reproduce a Reynolds-like behaviour (Reynolds, 1987) on a swinglet team. According to this framework, each member of a swarm adjusts his position at any given time according to three rules: a) collision avoidance, b) velocity matching, and c) flock centering. The task is complicated by the fact that, in agreement with the recent line of research pursued at the EPFL, this work will assume that no GPS information is available to the MAVs. For this reason the aircraft will have to rely on alternative methods in order to detect their positions relatively to the other members of the swarm they are part of. Looking more in details at each of the main components of Reynolds' behaviour, the following considerations can be made:

a) without the possibility of relying on GPS information, collision avoidance could be implemented either installing some sensors on the MAVs (potential candidates are IR distance sensors[7], and ultrasonic range sensors[8]), or with an alternative approach. Options could be either "pinging" (i.e., sending special TCP/IP-like packets) other MAVs and use the success rate/time of response as an estimation of the distance, or measuring the strength of the connections between the various aircraft (maybe creating a Wi-Fi link among all of them and measuring the strength of the wireless signal). The latter approach could also be seen as a roughly replication of the olfactory system used by animals (despite generally it does not apply to birds).

b) the presence of an onboard autopilot makes possible to instruct all the MAVs to fly at a specific and constant speed during their entire flight. In this way, the issue of velocity matching could be easily tackled. Of course, robots interacting in a real physical environment, will be subjected to different forces (i.e., wind, air resistance, etc.) affecting their speed. The speed control system must therefore incorporate some "reactive" strategy. Furthermore, according to Reynolds' definition, "velocity" also includes heading. Within the setup discussed herein the MAVs could simply share with each other this information (gathered by their embodied sensors). In this case the flock might require a "leader" to determine the overall direction the group has to follow.

3) concerning flock centering, in his paper Reynolds defines this principle as follows:

> "*Flock centering makes a boid want to be near the center of the flock. Because each boid has a localized perception of the world. "center of the flock" actually means the center of the nearby flockmates. Flock centering causes the boid to fly in a direction that moves it closer to the centroid of the nearby boids. if a boid is deep inside a flock, the population density in its neighborhood is roughly homogeneous; the boid density is approximately the same in all directions. In this case, the centroid of the neighborhood boids is approximately at the center of the neighborhood, so the flock centering urge is small. But if a boid is on the boundary of the flock, its neighboring boids are on one side. The centroid of the neighborhood boids is displaced from the center of*

[7] For example: http://www.trossenrobotics.com/sharp-ir-distance-sensor-gp2y0a02yk.aspx
[8] For example: http://www.trossenrobotics.com/parallax-ping-ultrasonic-range-sensor.aspx

the neighborhood toward the body of the flock. Here the flock centering urge is stronger and the flight path will be deflected somewhat toward the local flock center."

The lack of GPS information makes the implementation of this point the most interesting part from a scientific point of view. A map of the geographical displacement of the entire swarm could be built dynamically by the MAVs relying on the estimated distances elaborated according to the principles outlined in 2). Further thinking and investigations are nonetheless required in order to address this issue.

The second point, target reaching behaviour, will involve a MAV stationary on the ground - acting as a target - and a MAV team flying over the area in which the target is deployed attempting to localise it. The target will act as a transmitting station, broadcasting a signal the other MAVs are able to perceive. A certain amount of aircraft will take off from the same position and then will distribute themselves among a certain area. The MAVs will independently look for the target, trying to get as close as possible to it. When one of the searchers has arrived close enough, it will communicate with the teammates (the establishment of a common communication protocol to be used will be left to an evolutionary process) in order to make all of them converging to the proper area.

REFERENCES

Aung, M., Purcell, G.H., Young, L.E., Amaro, L.R., Srinivasan, J., Ciminera, M.A., and Chong, Y.J., Autonomous Formation-Flying Sensor for the StarLight Mission (IPN Progress Report 45-152)

Christensen, A.L., and Dorigo, M., Incremental Evolution of Robot Controllers for a Highly Integrated Task (From Animals to Animats 9: Proceedings of the 9th International Conference on Simulation of Adaptive Behaviour, pp. 473-483, 2006)

Hauert, S., Leven, S., Zufferey, J.-C., and Floreano, D., Communication-based Leashing of Real Flying Robots (Proceedings of ICRA 2010, IEEE International Conference on Robotics and Automation, in press)

Leven, S., Zufferey, J.-C., and Floreano, D., A Minimalist Control Strategy for Small UAVs (Proceedings of IROS 2009, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2873-2878, 2009)

Lissaman, P.B.S., and Shollenberger, C.A., Formation Flight of Birds (Science, 22(168), no. 3934, pp. 1003–1005, 1970)

Naffin, D.J., and Sukhatme, G.S., A Test Bed for Autonomous Formation Flying (Institute for Robotics and Intelligent Systems Technical Report IRIS-02-412, 2002)

Petrovic, P., Overview of Incremental Approaches to Evolutionary Robotics (Proceedings of the Norwegian Conference on Computer Science, pp. 151-162, 1999)

Reynolds, C.W., Flocks, Herds, and Schools: A Distributed Behavioral Model (Computer Graphics, 21(4), pp. 25-34, 1987)

Ruini, F., and Cangelosi, A., An Evolutionary Robotics 3D model for autonomous MAVs navigation, target tracking and group coordination (Proceedings of IJCNN 2010, International Joint Conference on Neural Networks, in press)

Ruini, F., and Cangelosi, A., Extending the Evolutionary Robotics Approach to Flying Machines: an Application to MAV Teams" (Neural Networks, no. 22, pp. 812-821, 2009)

Ruini, F., Cangelosi, A., and Zetule, F., Individual and Cooperative Tasks performed by Autonomous MAV Teams driven by Embodied Neural Network Controller (Proceedings of IJCNN 2009, International Joint Conference on Neural Networks, pp. 2717-2724, 2009)

Tillerson, M., Breger, L., and How, J.P., Distributed Coordination and Control of Formation Flying Spacecraft (Proceedings of IEEE American Control Conference, pp. 1740-1745, 2003)

Weimerskirch, H., Martin, J. Clerquin, Y., Alexandre, P., and Jiraskova, S., Energy saving in flight formation (Nature, 413, pp. 697-698, 2001)

Flocking behaviours in Micro-unmanned Aerial Vehicles: towards experiments on real robots

Fabio Ruini[1*] and Angelo Cangelosi[1]

[1]Centre for Robotics and Neural Systems, School of Computing and Mathematics, University of Plymouth, UK

15 May 2011

Abstract

This report illustrates the research efforts covered by the EOARD grant that have been conducted over the last few months. Rather than concentrating on computer simulations exclusively - as done for the work carried out previously - the authors have now directed their main research focus toward the usage of real robotics Micro-unmanned Aerial Vehicle (MAVs). The testbed platforms used as reference is the *swinglet*, a lightweight mono/fixed-wing robotic aircraft produced by the Swiss company senseFly^{TM}.

The report will first illustrate the most relevant technical details about the MAV used, then will provide an overview of the computer model elaborated for designing and testing autonomous controllers for the *swinglet*. The algorithms developed on the software simulator implement both individual/collective navigation (using approximated areas of attraction or geographically accurate waypoints) and flocking (speed adjustment, heading alignment, Reynolds' boids-like) functionalities.

The work described in this report has also been made possible thanks to a collaboration with the Laboratory of Intelligent Systems (LIS) at the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. The authors would like to thank again anyone involved in this joint effort.

*To whom correspondence should be addressed. Email: fabio.ruini@plymouth.ac.uk.

1 Robotics platform used: senseFly's swinglet

The research presented herein has been carried out using a customised version of the *swinglet*[1] (see Figure 1), a 420g light 80cm wing-span mono/fixed-wing MAV produced by senseFly™[2], generally used for aerial photography/surveillance and scientific investigations on outdoor flying robots [1]. Its main structure is made of expanded polypropylene, and it mounts an electric propeller fuelled by a polymer lithium battery guaranteeing up to 60 minutes of autonomy (enough to approximately cover a 40km distance). According to the specifications provided by the manufacturer, the *swinglet* flies at a speed included between 10 and 15m/sec, has a maximum turnrate of 45°/sec (guaranteed by the use of two elevons[3], one on each side of the aircraft) and can proficiently cope with wind currents as strong as 25km/h.

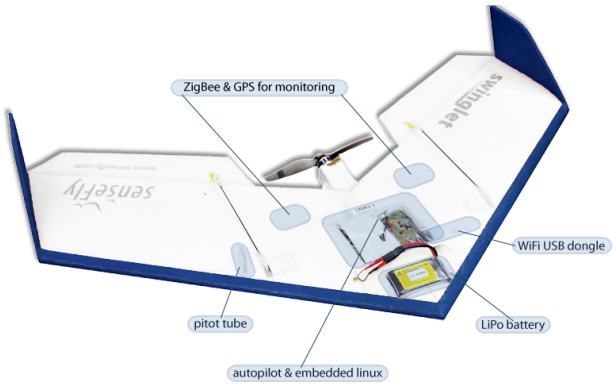


Figure 1: senseFly's *swinglet* Micro-unmanned Aerial Vehicles

The configuration we have access to comprises of a GPS receiver (namely a u-blox™LEA-5H GPS module[4]), a rate gyroscope (Analog Devices™ADXRS610[5]) used to measure the absolute yaw rate of the MAV, and two pressure sensors (implemented as pitot tubes and belonging to the Freescale™MPX series[6]). In order to exchange data with a ground-based station the *swinglet* is also equipped with a Digi™XBee-PRO PKG-U[7], a radio transmitter providing a 1.6km wide communication range. Standard R/C equipment is used for actuators, motor controller and battery.

[1] `http://www.sensefly.com/products/swinglet/`
[2] senseFly (`http://www.sensefly.com`) is an EPFL spin-off company.
[3] Elevons are combined ailerons/elevators.
[4] `http://www.u-blox.com/en/lea-5h.html`
[5] `http://www.analog.com/static/imported-files/data_sheets/ADXRS610.pdf`
[6] `http://www.freescale.com/webapp/sps/site/taxonomy.jsp?nodeId=01126990368716`
[7] `http://ftp1.digi.com/support/documentation/90000831_A.pdf`

A complete autopilot system has been built by Leven et al. [2] and installed on the aircraft through a dsPic33 micro-controller[8] (see Figure 2(a)). The approach followed by the designers of this system is different than those usually undertaken for implementing autopilots. Leven's technique can be considered 'minimalist' since it only relies on two pressure sensors and a single axis rate gyro, rather than on a complete IMU (Inertial Measurement Unit[9]) or an AHRS (Attitude and Heading Reference Systems[10]) as it is common habit among the experts of the field (see for example [3] and [4]).

The autopilot has direct reading access to all the sensors mounted on the aircraft and it can control both the propeller thrust and the servomotors that in turn lower/raise the elevons. In addition to flight stabilisation the system can therefore perform control of airspeed, altitude and heading turnrate. Some basic autonomous navigation functions, such as waypoint-based ones, are already implemented within the system. Furthermore the autopilot provides an automatic landing function based on GPS, which - either on request or in case of a software/hardware failure - forces the MAV to fly toward a pre-specified landing spot, then makes it glide around it progressively reducing its altitude and the thrust of the propeller until the aircraft gets in touch with the ground.

A flexible payload bay situated on the top surface of the aircraft - next to the battery compartment - allows the *swinglet* to transport up to 150g of additional payload. In the available configuration the payload consists of both the above mentioned autopilot system, and a Toradex^{TM}Colibri PXA270 CPU board[11] running a minimal Linux distribution. To the board is connected an off-the-shelf USB Wi-Fi dongle (a dual band Wireless-N Netgear^{TM}WNDA3100[12]) which can be used for communication between the MAVs. The dongle has a 500m line-of-sight communication range, although its firmware has been modified in order to allow the experimenter to restrict this range as desired [5]. The onboard computer is directly connected to the autopilot (see Figure 2(b)), to which it can issue commands (namely desired turnrate, speed and altitude) via a software controller running on the CPU board.

Thanks to the XBee radio link, the MAV behaviour can be monitored by a

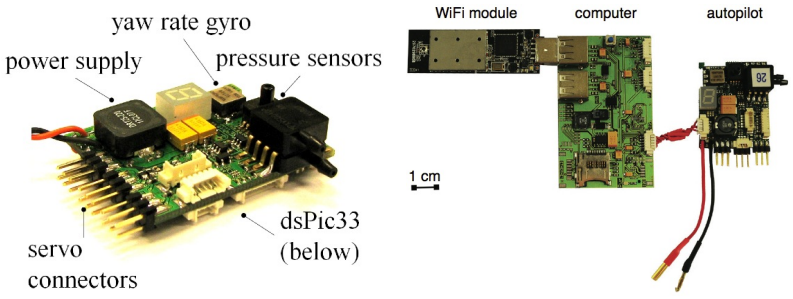
[8] `http://www.microchip.com/ParamChartSearch/chart.aspx?mid=14&lang=en&branchID=8183`

[9] *An IMU is an electronic device that measures and reports on a craft's velocity, orientation, and gravitational forces, using a combination of accelerometers and gyroscopes* (from: `http://en.wikipedia.org/wiki/Inertial_measurement_unit`).

[10] *An AHRS consists of sensors on three axes that provide heading, attitude and yaw information for aircraft. They are designed to replace traditional mechanical gyroscopic flight instruments and provide superior reliability and accuracy* (from: `http://en.wikipedia.org/wiki/Attitude_and_Heading_Reference_Systems`).

[11] `http://www.toradex.com/Products/Colibri/Modules/Colibri_PXA270_312MHz`

[12] `http://www.netgear.co.uk/wnda3100.php`



(a) (b)

Figure 2: (a) the dsPic33 micro-controller upon which the autopilot has been built; (b) overall view of all the equipment hosted inside the payload bay; from left to right: the USB Wi-Fi dongle, the Linux board, and the micro-controller/autopilot

standard computer located on the ground and running a dedicated software called *e-mo-tion* (see a screenshot in Figure 3) developed by Beyeler et al [6]. The computer simply needs to be plugged to a proper XBee^{TM}device[13] capable of exchanging data with the radio unit installed on the aircraft in order to be used as ground station. *e-mo-tion* allows the user to switch the control of the aircraft between the auto-pilot[14], the software controller running on the Linux board and a standard radio transmitter (in case of necessity the *swinglet* can be remotely controlled, which is obviously an extremely useful property to rely on during testing). Another interesting feature offered by the software running on the ground station is the possibility of logging all the flight data, thus relieving the controller operating on the Linux board from this task. In this way the code running on the onboard computer will be lighter (also possibly less bug-prone), and the user will not incur in the risk of filling the flash memory on the embedded computer, thus potentially avoiding unexpected crashes.

2 Software simulator

The software simulator used for the preliminary testing of the controllers described herein (see Figure 4) is a modified version of the one described in [7, 8]. As before the simulator implements an incremental geometric flight model in discrete time-steps [9]. The parameters of the model have been tuned in order to replicate in the

[13]Each XBee^{TM}unit used on the ground-based station can connect up to a maximum of 3 MAVs.

[14]Through *e-mo-tion* is also possible to interact with the autopilot, for example deploying specific waypoints MAV will have to follow, or forcing a landing procedure.

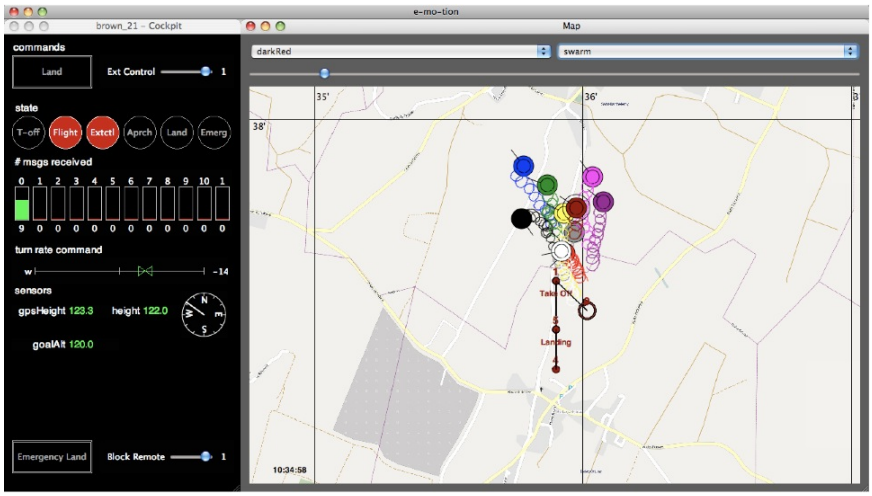


Figure 3: Screenshot of the *e-mo-tion* main interface during a flight involving several MAVs

most accurate way possible the constraints of the senseFly^{TM}'s *swinglet* platform, specifically in terms of speed range and turn rate per second.

The autonomous controllers driving the aircraft operate on two variables: airspeed (which can be increased or decreased) and turn-rate (which can be modified instructing the MAV to perform a yaw turn, i.e. a rotation around its vertical body axis). The reason for reducing the controller to operate on these two dimensions only relies on the autopilot system described within the previous section. As noted before, the autopilot provides both flight stabilisation and altitude control (other than being able to modify speed and turnrate), meaning that the autonomous controller can assume the MAV being always parallel to the ground and flying at the desired altitude, thus ignoring aspects as current pitch and roll angles. In this way the MAVs can be considered to some extent sorts of "2D flying robots", since their behaviour will only depend on rotations around one single axis, as it is generally the case for ground-based vehicles.

The virtual reference environment implemented in this version of the simulator consists of a three-dimensional parallelepiped measuring 800x300x800GUs (where the height is represented as the X axis; see Figure 5, which also highlights the coordinate systems used by Irrlicht and therefore in the rest of this report)[15].

The MAVs - having a size of 4.35x1.797x4.82GUs - can fly across the environment at a speed included between 10 and 15GU/time-step. At any time-step the

[15]We use the term GU to indicate 'Graphical Unit', i.e. the basic measure unit employed by Irrlicht graphics engine. Efforts have been spent to obtain the following two relationships: 1GU≈1m, 1time-step≈0.1sec. Differently than the previous simulators, in this version the environment boundaries can be stepped over with no consequences for the MAVs.

Figure 4: Screenshot of the software flocking simulator

MAVs first perform a turn-rate (if so decided by the controller, in which case the rotation must be included within a $[-4.5°; 4.5°]$ range) then they are all moved in sequence.

Each aircraft moves along its current heading (after the yaw preliminary rotation) redeploying itself at a distance calculated according to Equation 1 (where i is used as a general index for indicating a non-specific MAV).

$$distance = \frac{MAV_i.speed + X \sim (0, 0.25)}{10} GUs \tag{1}$$

The new coordinates are calculated as in Equation 2 (please consider that the addition and the multiplication operations have to be interpreted as vector addition and scalar multiplication respectively).

$$\vec{MAV}_i^{t+1} = \vec{MAV}_i^{t} + distance * \vec{transformationVector} \tag{2}$$

transformationVector is a three-dimensional vector, for which the X, Y, and Z elements are defined as specified in Equation 3[16]:

$$\begin{aligned} X &= cos(MAV_i.\hat{x}) * sin(MAV_i.\hat{y}) * cos(MAV_i.\hat{z}) + sin(MAV_i.\hat{x}) * sin(MAV_i.\hat{z}) \\ Y &= cos(MAV_i.\hat{x}) * sin(MAV_i.\hat{y}) * sin(MAV_i.\hat{z}) - sin(MAV_i.\hat{x}) * cos(MAV_i.\hat{z}) \end{aligned} \tag{3}$$

[16]These operations are performed within the software by the dedicated Irrlicht functions *setRotationDegrees()*, *setRotationRadians()*, and *transformVect()*.

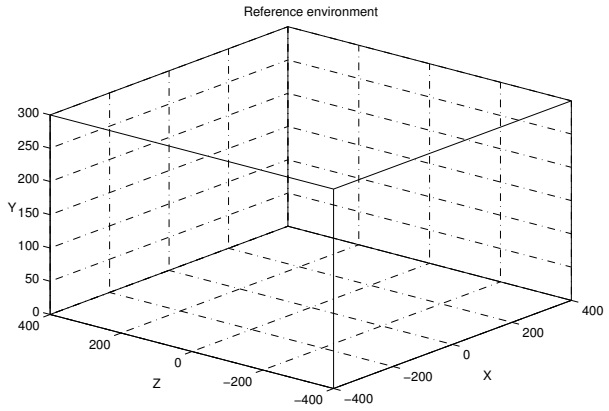


Figure 5: Simulation reference environment

$$Z = cos(MAV_i.\hat{x}) * cos(MAV_i.\hat{y})$$

A certain amount of noise - distributed according to a Gaussian distribution with mean 0 and standard deviation 0.25 (what in Equation 1, as well as in the next paragraphs, is defined as $X \sim (0, 0.25)$) - is added to both any yaw manoeuvre and forward movement performed by the MAVs simulated through this computer model. The reason for introducing noise consists in adding some degrees of uncertainty to the simulated flight dynamics, thus increasing the realism of the model and the robustness of the controllers tested in it [10].

The simulator allows the user to set several parameters before running an experiment[17]:

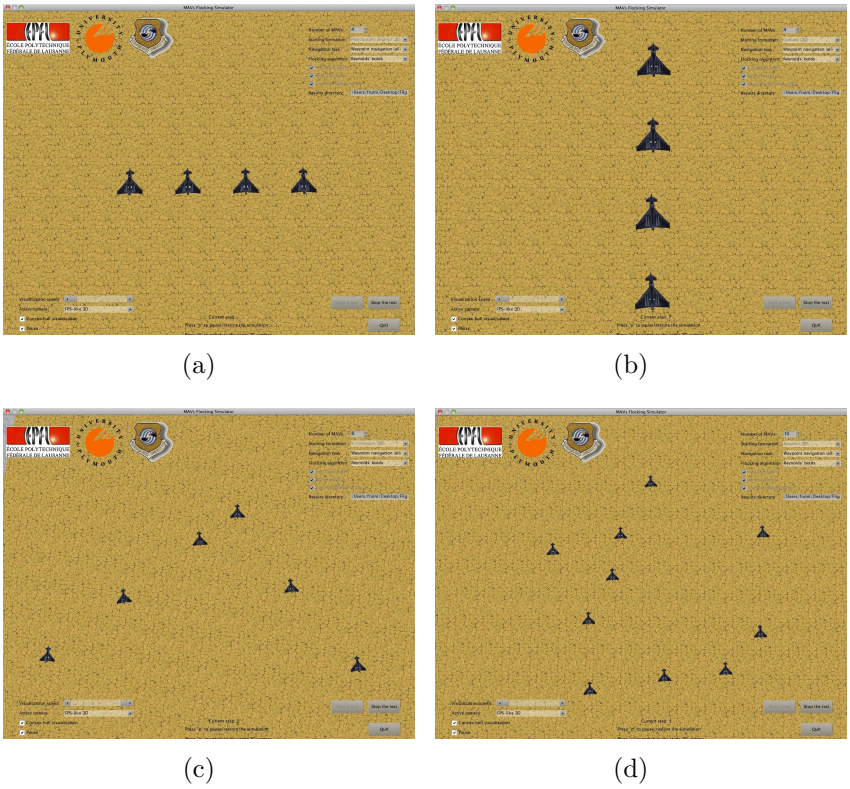
- *number of MAVs*: from a minimum of 1 up to a maximum of 12;
- *initial MAVs team formation*: horizontally aligned (2D), queued (2D), V-formation (2D), intervallic launch (2D), random (2D), horizontally aligned (3D), queued (3D), V-formation (3D), intervallic launch (3D), random (3D);
- *navigation task*: none, fly around the centre of the environment (leader only), fly around the centre of the environment (entire team), waypoint navigation (leader only), waypoint navigation (entire team), follow the leader (who flies around the centre of the environment), follow the leader (who flies between two waypoints);
- *flocking algorithm*: none, speed adjustment, heading alignment (to the leader's heading), heading alignment (to the average neighbours' heading), heading

[17]Some of these parameters, as the navigation task and the flocking algorithm used, can also be modified in real time while a simulation is running.

alignment (to the leader's heading) + speed adjustment, heading alignment (to the average neighbours' heading) + speed adjustment, Reynolds' boids.

2.1 Initial formation

The initial formation parameter permits to select the way in which the MAVs will start each test. The two main categories the user can choose from are 2D and 3D. The former means that the aircraft will all be flying at the same altitude, while the latter deploys the MAVs at different altitudes.



(a) (b)

(c) (d)

Figure 6: Different initial MAV teams formations: (a) 4 MAVs horizontally aligned; (b) 4 MAVs queued; (c) 6 MAVs reproducing a V-formation; (d) 10 MAVs randomly distributed

When horizontal alignment or queueing are selected (respectively shown in Figure 6(a) and 6(b)), the MAVs are respectively deployed side by side or forming a queue, standing in both cases at 10GUs (d) distance far from each other[18]. V-

[18]In case of a 3D initial formation the distances are calculated considering the MAVs all being at the same altitude.

formation and random deployment are implemented according to Algorithms 1 and 2 (where: h indicates the desired altitude; N stands for the number of MAVs within the team; *areMAVsTooClose()* is a function that checks whether in the group there are two or more MAVs too close to each other, i.e. within a distance $< d$ between them; *randFloat()* is a function returning an uniformly distributed random float value included between the lower and upper boundaries specified in input)[19].

Algorithm 1 V-formation MAVs deployment (3D)

$d = 10$;
$MAV_1.x = 0; MAV_1.y = h; MAV_1.z = 0$;
for i=2:N **do**
  $MAV_i.y = h + randFloat(-5.0, 5.0)$;
  $x = d * i; z = d * i$;
  if $i\%2 = 0$ **then**
    $MAV_i.x = MAV_1.x - x$;
    $MAV_i.z = MAV_1.z - z$;
  else
    $MAV_i.x = MAV_1.x + x$;
    $MAV_i.z = MAV_1.z + z$;
  end if
end for

Algorithm 2 Random MAVs deployment (3D)

$d = 10; d_2 = N \div 2.5$;
while areMAVsTooClose() **do**
  for i=1:N **do**
    $MAV_i.x = randFloat(-d * d_2, d * d_2)$;
    $MAV_i.y = h + randFloat(-5.0, 5.0)$;
    $MAV_i.z = randFloat(-d * d_2, d * d_2)$;
  end for
end while

The two methods defined as 'intervallic' launches indicate that the MAVs are sequentially deployed into the reference environment according to their ID[20]. In

[19] Algorithms 1 and 2 refer to the 3D scenarios. The altitude (y axis) is simply set equal to h in case a 2D deployment method is selected.

[20] When the MAV objects are generated by the software, each of them receives an ID number starting from 1 and increasing sequentially. The upper boundary of the IDs range is N, which correspond to the number of MAVs used.

the 2D scenario the aircraft just appear at the centre of the environment with a certain time delay between each of them. Instead - in the 3D scenario - the MAVs take off in series from the ground, then follow a fixed path which brings them at the desired altitude through progressive modifications of their pitch rate (see Figure 7). In both cases, the interval between each "launch" amounts to 500 time-steps.

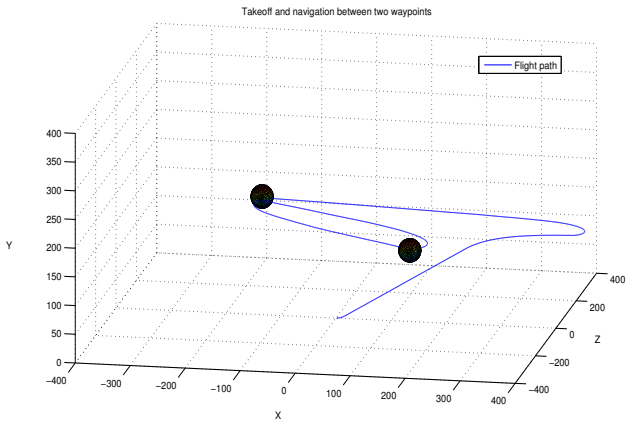


Figure 7: Flight path followed by a single MAV taking off from the ground and then navigating between two waypoints

2.2 Navigation algorithm

Concerning navigation, the MAVs - as mentioned before - can be driven by three different categories of algorithms: *a)* fly being attracted to the centre of the environment, *b)* navigate back and forth between different (fixed) waypoints, or *c)* follow a 'leader' teammate.

To implement the attraction toward the centre of the environment the simple formula expressed in Equation 4 - which return a steering request - has been used. The numerator of this formula computes the distance from the centre of the environment for the i-th MAV (in two-dimensions only, which is the reason for omitting the $MAV_i.y^2$ term). The denominator simply provides to calculate the diagonal length for the base of the reference environment, then divides the obtained value by 2. Gaussian noise is added to the resulting steering request.

$$steeringRequest = \frac{\sqrt{MAV_i.x^2 + MAV_i.z^2}}{(\sqrt{800^2 + 800^2})/2} + X \sim (0, 0.25) \tag{4}$$

Figure 8 shows, in two-dimensions, the flight paths that 4 aircraft have followed during a simulation they started deployed according to a random 2D formation.

The MAVs have flown for about 10,000 time-steps while being attracted toward the centre of the environment.

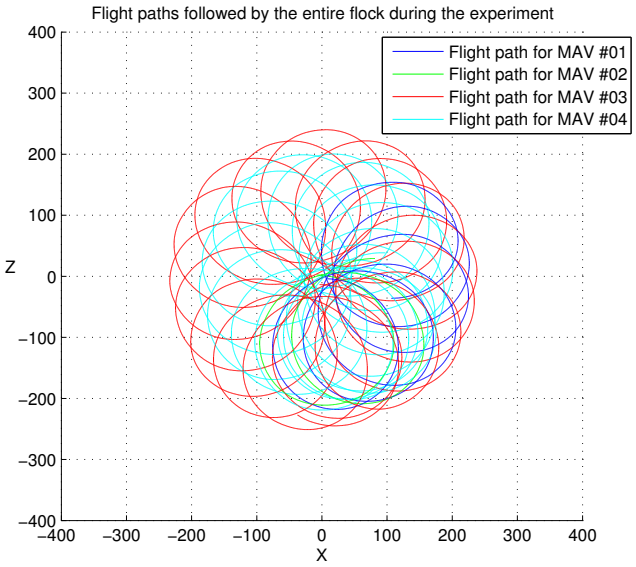


Figure 8: Two-dimensional flight paths followed by a team of 4 MAVs attracted to the centre of the reference environment. Their paths describe a series of circles - moving counterclockwise - passing through the central point

Waypoint navigation simply consists in the MAV flying between two fixed points in the space. The two waypoints used are respectively located at coordinates $(-165.0, h, 165.0)$ and $(165.0, h, -165.0)$, thus 300GUs far from each other[21] (h, within this context, represents the altitude of the all MAVs or the altitude of the MAV with the lowest ID for 2D and 3D starting formations respectively). The steering request generated by the controller at any time-step is calculated according to Equation 5. In this equation the $\Delta\alpha$ symbol indicates the angle between the current waypoint and the heading of the MAV (assuming the aircraft being parallel to the ground, i.e. with a roll/bank angle equal to 0°); its value falls within the $[-180.0°, 180.0°]$ range ($[-180.0°, 0.0°]$ when on the relative "left" of the MAV, $[0.0°, 180.0°]$ when on its relative "right"). Again, normally distributed noise is added to the value generated by the equation to allow for some uncertainty in the outcome of the executed manoeuvre. A waypoint is considered reached when one of the MAVs gets closer than 30GUs to it. The two waypoints, as well as the

[21]The Euclidean distance between two points a and b both laying inside the same three dimensional space can be calculated according to the following equation, where (x_a, y_a, z_a) are the coordinates of point a, and (x_b, y_b, z_b) those of point b: $d = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2}$.

trajectory followed during a test by a MAV navigating between them (inclusive of the take off phase), can be seen in Figure 7.

$$steeringRequest = \frac{\Delta\alpha}{40} + X \sim (0, 0.25) \tag{5}$$

The simulator allows to make all of the MAVs fly between waypoints/around the centre of the environment, or just one of them (the "leader") obeying to such a navigation task. If the latter feature has been activated, the non-leader MAVs will have two options available (depending on the choice made by the user): not doing anything (i.e., flying on a straight line), or fly following the leader. When an algorithm (whether a navigation or, as we will see in the next section, a flocking one) which involves a leader is selected, the designed one to assume that role is the MAV with the lowest assigned ID.

Leader-following behaviour and waypoint navigation have been implemented in a very similar way. When one of the two navigation algorithms belonging to the leader-following category is selected, the leader either navigates attracted to the centre of the environment or flies back and forth the two fixed waypoints. The followers are driven by steering requests generated through Equation 5, with the only difference consisting in the calculation of the $\Delta\alpha$ parameter which now represents the angle between the follower's heading and the position of the leader, rather than the angular difference between the heading direction of the follower and the current waypoint.

A few more notes must be added before closing this section:

- for 3D intervallic launches the navigation algorithm only kicks off once the MAV has reached the desired altitude;
- common across all the navigation algorithms is the fact that the MAVs start every simulation flying at a speed equal to 12GU/time-step.

2.3 Flocking algorithms

The software simulator permits to do not assign any navigation task to the aircraft. In this case all the MAVs will simply go straight, unless a flocking algorithm which from time to time force them to steer has been selected. Flocking algorithms are so labeled because they aim to make the MAV team behave like a flock. Various alternative strategies have been tested to obtain this outcome and a description of how they work is provided within the current section.

From a technical point of view, the flocking algorithms that generate a steering requests[22] make the MAV perform, at any time-step, a yaw rotation which is the

[22]Only one of them, namely *Speed adjustment*, does not generate any steering request at all.

sum (intended as sum of circular quantities) of two independent steering requests: one coming from the navigation algorithm (if enabled), the other one from the flocking rule.

The first option available to the MAVs is speed adjustment. According to this algorithm, which only works when waypoint navigation is used[23], the designated leader continuously broadcasts information about its actual coordinates and those of the waypoint he is currently aiming to. The other MAVs receive this information in real time[24] and elaborate it according to Algorithm 3 (where: *distance()* is a function which returns the distance between the two points specified as input parameters, *flockingDistance* is the desired distance at which the followers should stay far from the leader, MAV_0 is the flock leader, and $waypoint_j$ is the waypoint toward which the leader and/or all the MAVs is/are currently flying). As the algorithm shows, a 2GUs tolerance has been added to the calculus of the desired flocking distance.

Algorithm 3 Speed adjustments for the i-th non-leader MAV

```
flockingDistance = 10;
if distance(MAV_i, waypoint) > distance(MAV_0, waypoint) then
  if distance(MAV_i, MAV_0) > (flockingDistance + 2) then
    MAV_i.speed++;
  else if distance(MAV_i, MAV_0) < (flockingDistance − 2) then
    MAV_i.speed−−;
  end if
else
  MAV_i.speed−−;
end if
```

This simple code allows the MAVs to infer their relative position compared to the leader (i.e., whether they are in front of or behind it, "understood" exploiting the knowledge about the current waypoint coordinates) and then adjust their speed accordingly (increasing it if behind the leader and farther than the desired flocking distance, decreasing it if in front of the leader or too close to it). The intensity of the speed adjustment corresponds to a random float value drawn from a flat distribution ranging between 0 and 0.5.

[23]This algorithm could potentially work with "attraction to the centre" as navigation task, but this functionality has not been implemented in the simulator.

[24]Please consider that within the simulator all the MAVs have instantaneous access to all the information they need. As we will see in next sections, this is not true for real robots since all the required information must be exchanged between the robots, thus leading to some delay in communication and to the necessity of staying within a limited distance range.

An additional flocking algorithm implemented in the simulator is heading alignment. As the name suggests, this algorithm provides to modify the heading of each MAV in order to match it with a reference one. Working as references can be either the leader's heading (as usual, considered as 'leader' is the MAV associated with the lowest ID, independently from the navigation algorithm in use[25]) or the average heading for all the MAVs within the neighbourhood.

Once the $\Delta\alpha$ between the current heading and the desired one has been calculated, the amount of steering to perform is elaborated according to Equation 5. In algorithmic terms the entire procedure can be represented by the pseudocode in Algorithm 4, where: *calculateDeltaHeadingFromTo()* is a function which return the angle between the heading of the object received in input (first parameter), and a different object for whom the coordinates are received by the function as second parameter; *calculateAverageNeighboursHeading()* is a function calculating the average neighbours' heading for the MAV specified in input.

Algorithm 4 Heading alignment for the i-th MAV

$\Delta\alpha = 0$;
if $flockingAlgorithm = alignToTheLeader$ **then**
  if $i \neq 0$ **then**
    $\Delta\alpha = calculateDeltaHeadingFromTo(MAV_i, MAV_0)$;
  end if
else if $flockingAlgorithm = alignToTheNeighboursHeading$ **then**
  $avgHeading = calculateAverageNeighboursHeading(MAV_i)$;
  $\Delta\alpha = calculateDeltaHeadingFromTo(MAV_i, avgHeading)$;
end if
$performYaw(MAV_i, \Delta\alpha \div 40 + X \sim (0, 0.25))$;

The simulator also allows to use mixed flocking algorithms in which speed matching and heading alignment (i.e., Algorithm 3 and 4) are used together. Futrhermore the user can decide to employ the Reynolds' flocking algorithm for boids. What this algorithm is and how does it work will be explained in details in the next two sections.

Please consider that also all of the flocking algorithms generate an output steering manoeuvre which is affected by Guassian distributed noise.

2.3.1 Reynolds' flocking algorithm for boids

With the term *Reynolds' algorithm* we refer to the core of the software Craig Reynolds originally designed to implement automatic flocking behaviour among

[25]In case this flocking algorithm is selected it will not have any impact on the leader, since it is not supposed to perform any steering manoeuvre in order to match its own heading.

computer animated agents (or *boids*, i.e. birds-like objects, according to his definition). His work - originally thought in order to help the workers in computer graphics involved in designing the motion of large groups of entities - eventually led in 1987 to a seminal publication introduced at the annual edition of the SIGGRAPH[26] conference [9]. The work presented by Reynolds received a great deal of attention quickly becoming extremely popular within the computer science field[27].

What is remarkable in Reynolds' work consists in the assumption upon which his model is based. Rather than elaborating complex rules to govern the behaviour of a flock considered as a whole, he proposed an approach based on every single individual obeying to a limited set of pretty simple rules. Other than simple these rules are also local, in the sense that every boid is only aware of its local neighbourhood (i.e., the behaviour exhibited by the boids closer to it than a certain threshold) and does not have access to global information about the entire flock at all. This mechanism has proven to be working. Large groups of boids driven by Reynolds' algorithm are capable of showing a coherent flocking behaviour, as well as higher level properties such as collective obstacle avoidance[28]. Further improvements that have been made over the years on top of the original algorithm allow for a flock to be directed (even if that would mean losing one of the characteristic traits of the model, which is the absence of global information[29]) [12], to replicate leader-following dynamics [13], etc.. Modified versions of the algorithm have also been applied countless time to reproduce for example the motion of animals as in schooling [14] or herding [15].

Before discussing the technical details about the flocking algorithm it is worth to consider how, at the time he wrote his software, Reynolds was not interested in replicating real animal behaviour in computer animations, so he never claimed that his model faithfully recreates conducts that can be observed in nature. Fur-

[26]SIGGRAPH (International Conference and Exhibition on Computer Graphics and Interactive Techniques, `http://www.siggraph.org`) is a traditional conference dedicated to the computer graphics community that in 2011 will be held for the 38^{th} time.

[27]Scientists from apparently unrelated fields, as complexity science, also looked with interest at Reynolds' model, seeing in it an excellent demonstration of a complex collective behaviour emerging from the low-level interactions of a multitude of agents, each of them being aware of (as well as influencing) a narrow neighbourhood only. This is a classical example of what Murray has defined as the *molecular view* of complexity [11].

[28]To have an example of Reynolds-based flocking behaviour the reader can watch either a video about the basic behaviour elaborated by the author (`http://www.youtube.com/watch?v=2aXMo3MFNsA`) and one in which the boids: a) are attracted by a point moving across the space; b) at the same time need to avoid a fixed obstacle (`http://www.youtube.com/watch?v=GUkjC-69vaw`).

[29]Although it might be argued that not all of the boids need to be aware of this information. The modified algorithm would preserve Reynolds' assumption if only the boids in front of the group would be able to perceive the point of attraction and steer toward it, thus back-propagating a steering manoeuvre to the entire flock.

thermore, another assumption he implicitly made about all the members of a flock being peers (i.e., no hierarchical structures existing within the group) is not necessarily true (and it has recently been challenged by Biro and colleagues [16, 17, 18] in their studies focusing on homing pigeons).

The set of rules governing the flocking behaviour can be summarised (in order of decreasing importance) as:

1. *collision avoidance*: avoid collisions with nearby flockmates;
2. *velocity matching*: attempt to match velocity with nearby flockmates;
3. *flock centering*: attempt to stay close to nearby flockmates.

To fully understand how these three rules work, it is fundamental to remember that Reynolds defined "velocity" as "*a vector quantity, referring to the combination of heading and sepeed*". Rule number #2, velocity matching, therefore refers to both speed adjustment (match the speed of the boids inside the neighbourhood) and heading alignment. To clear up potential misunderstandings, in a later paper [12] Reynolds renamed the velocity matching rule as *alignment*. Collision avoidance and flock centering are instead two complementary rules that respectively provide to keep the boids at a 'safe distance' between each other, but not too far away (i.e., close enough to be consider a homogeneous group to the eyes of an external observer).

Every rule generates an independent request for a steering manoeuvre to be executed by the boid under examination. The entire model relies on vector geometry, implying that the steering requests generated by the different rules are expressed in terms of independent geometric vectors. But, considering that a boid can only perform one steering manoeuvre per time, how is it supposed to behave when facing rules that attempt to steer it toward opposite directions? To solve this potential source of troubles, Reynolds first assigned different weights to the three rules, thus attributing a different relative importance to each of them. Not only the vectors generated by the different rules are attenuated by a certain factor[30], but a "governing element", named *accumulator*, was also introduced. The working principle of the accumulator is pretty easy to understand. As Reynolds explained [9]:

> *The acceleration [steering] requests are considered in priority order and added into an accumulator. The magnitude of each request is measured and added into another accumulator. This process continues until the sum of the accumulated magnitudes gets larger than the maximum*

[30]The weights associated to the rules are fractional values included between 0 and 1.

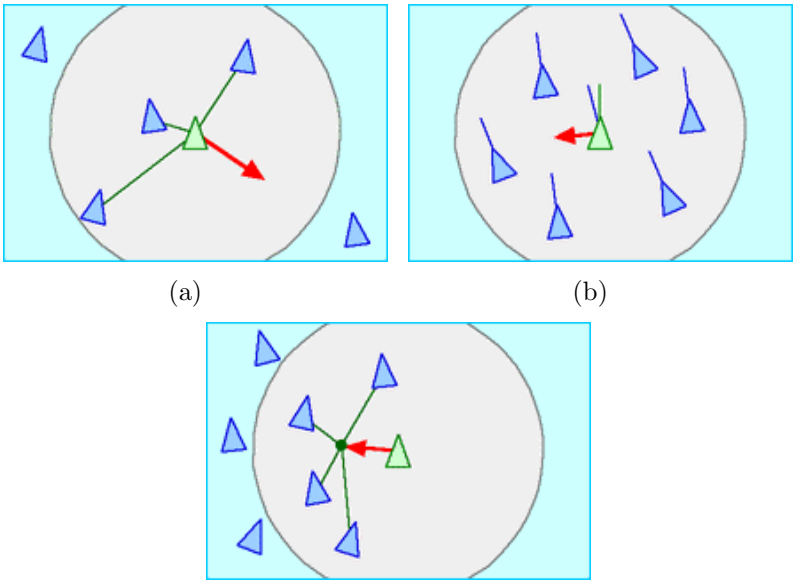


Figure 9: Graphics representation of the three rules elaborated by Reynolds: (a) separation; (b) alignment/velocity matching; (c) cohesion

> *acceleration [steering] value, which is a parameter of each boid. The last acceleration request is trimmed back to compensate for the excess of accumulated magnitude.*

Despite the entire procedure appears to be quite straightforward, Reynolds' original paper lacks in terms of technical details. Making tough to replicate its work in a faithful way.

2.3.2 Customised (Parker's-based) implementation of Reynolds' algorithm

As aforementioned there are some degrees of uncertainty about how Reynolds originally implemented his model. His original paper, although well written and capable of stimulating endless discussions, is quite minimalist in terms of technical details. It provides an overview of the general principles followed in order to achieve the flocking behaviour, but does not go particularly far in describing how exactly the entire procedure should be implemented in terms of computer code. For this reason, over the years many researchers have proposed their own implementations of Reynolds' algorithm. The approach we have decide to take inspiration from is

the one recently elaborated by Conrad Parker[31].

Parker's pseudocode for the separation and cohesion rules is shown in Algorithms 5 and 6 respectively. Consider that b_j represents the j-th boid belonging to the flock and N can be interpreted whether as the total amount of boids or just as the subset of those within j's neighbourhood.

Algorithm 5 Parker's pseudocode for flocking: cohesion rule

```
vector pc_j;
for b = 1 : N do
  if b ≠ b_j then
    pc_j = pc_j + b.position;
  end if
end for
pc_j = pc_j/(N − 1);
return (pc_j − b_j.position)/100;
```

Algorithm 6 Parker's pseudocode for flocking: separation rule

```
vector c = 0;
for b = 1 : N do
  if b ≠ b_j then
    if |b.position − b_j.position| < 100 then
      c = c − (b.position − b_j.position)
    end if
  end if
end for
return c;
```

We have decided to do without one of the three original Reynolds' rules (namely, speed matching), thus just relying on two of them. The reason for that is in using a point of attraction (waypoints, in our case) valid for all the MAVs, which would make redundant (whether not counterproductive) the employment of an alignment/velocity matching rule. Notwithstanding how many rules are employed, Algorithm 7 shows how all of these can be assembled together in order to generate a single steering manoeuvre and the consequent movement of boid b to the desired location. In that algorithm $m1, m2, m3, ...$ represent the weight factors applied to the vectors $v1, v2, v3, ...$ generated by rules 1, 2, 3, ... respectively. As can be seen

[31]The pseudo-code written by Parker, as well as a detailed explanation about the various assumptions he made in writing it, can be found online at the URL: `http://www.kfish.org/boids/pseudocode.html`.

Parker has not implemented any sort of accumulator, just relying on a weighted sum of all the vectors created by the individual flocking rules.

Algorithm 7 Parker's pseudocode for flocking: assembling the rules together

vector $v1, v2, v3, ...$;
int $m1, m2, m3, ...$;
boid b;
for $b = 1 : N$ **do**
  ...
  $v1 = m1 * rule1(b)$;
  $v2 = m2 * rule2(b)$;
  $v3 = m3 * rule3(b)$;
  ...
  $b.velocity = b.velocity + v1 + v2 + v3 + ...$;
  $b.position = b.position + b.velocity$;
end for

Algorithms 8 and 9 are simple translations of the algorithms detailed above with a syntax coherent to the one used in previous sections. *flockingDistance* is a variable indicating the desired minimum distance between two members of the flock, and *calculateDistance()* is a function returning the distance between the two objects specified in input. The aircraft for which the rules are computed is always assumed to be $MAVi$.

Because of the autopilot system described in Section 1 we only consider two dimensions (X and Y) while extrapolating the vectors from the flocking rules, assuming as a constant value the altitude at which the MAVs fly.

Algorithm 8 Parker's-inspired pseudocode for flocking: cohesion rule

$resVector.x = 0, resVector.y = 0$;
for j=1:N **do**
  if $j \neq i$ **then**
    $\vec{ruleVector} = \vec{ruleVector} + \vec{MAV_j}$;
  end if
end for
$\vec{ruleVector} = \vec{ruleVector} \div (N - 1)$;

Assembling the rules together has been done through the pseudocode shown in Algorithm 10, for which the parameters were fixed at the end of a trial-and-error procedure. *calculateCohesionVector(id)* and *calculateSeparationVector(id)* are two functions returning the vectors for the *id*-th MAV generated by the cohesion and the separation rules respectively (Algorithms 8 and 9). These two vectors are

Algorithm 9 Parker's-inspired pseudocode for flocking: separation rule

$ruleVector.x = 0, ruleVector.y = 0;$
for j=1:N **do**
  if $j \neq i$ **then**
    if $calculateDistance(MAV_i, MAV_j) < flockingDistance$ **then**
      $\vec{ruleVector} = \vec{ruleVector} - \vec{MAV_i} - \vec{MAV_j};$
    end if
  end if
end for

summed to the one representing the current position of MAV_{id} in order to obtain a vector which identifies the desired position toward which the MAV should aim. Since the modelled aircraft is a fixed wing one (and cannot therefore move to the destination point ignoring its current orientation), out of this data it must be extrapolated the delta angle ($\Delta\alpha$) between the current heading and the desired vector. $\Delta\alpha$ is first divided by 40 - as for Equation 5 - then Gaussian distributed noise is added. On top of that, the resulting value (which is the amount of yaw steering the MAV has to perform) is further divided by 3 because of the observations extrapolated by our experiments, that seem suggesting how the flocking behaviour becomes more efficient when the steering value is supplementary attenuated.

Algorithm 10 Parker's-inspired pseudocode for flocking: assembling the rules together

$\vec{v1} = \frac{1}{1} * calculateCohesionVector(MAV_i);$
$\vec{v2} = \frac{1}{50} * calculateSeparationVector(MAV_i);$
$\vec{resultingVector} = \vec{MAV_{id}} + \vec{v1} + \vec{v2}$
$\Delta\alpha = calculateDeltaHeadingToResultingVector(MAV_i);$
$performYaw(MAV_i, (\Delta\alpha \div 40 + X \sim (0, 0.25)) \div 3));$

3 Conclusions and future work

This report has illustrated the work that has been carried out over the last few months. A software simulator specifically targeted to the senseFly^{TM}'s *swinglet* has been developed and few navigation and flocking algorithms have been tested. Thus extensive simulations have not been carried out yet, the preliminary experiments done using the computer model have shown the feasibility of the algorithms designed.

The next and final step of this research, scheduled for the immediate future, will first involve the definition of metrics useful to evaluate the flocking behaviours. Then the porting and testing of the above algorithms to the real robots will take place.

References

[1] S. Leven, J.-C. Zufferey, and D. Floreano, "A simple and robust fixed-wing platform for outdoor flying robot experiments," *Proceedings of the International Symposium on Flying Insects and Robots*, pp. 69–70, 2007.

[2] S. Leven, J.-C. Zufferey, and D. Floreano, "A minimalist control strategy for small uavs," *Proceedings of IROS 2009, the IEEE/RSJ International Conference on Intelligent RObots and Systems*, pp. 2873–2878, 2009.

[3] J.-H. Kim, S. Wishart, and S. Sukkarieh, "Real-time navigation, guidance, and control of a uav using low-cost sensors," *Proceedings of FSR 2003, the International Conference on Field and Service Robotics*, pp. 95–100, 2003.

[4] D. Kingston and R. Beard, "Real-time attitude and position estimation for small uavs using low-cost sensors," *Proceedings of the AIAA 3rd Unmanned Unlimited Systems Conference and Workshop*, 2004.

[5] S. Hauert, S. Leven, J.-C. Zufferey, and D. Floreano, "Communication-based swarming for flying robots," *Proceedings of the Workshop on Network Science and Systems Issues in Multi-Robot Autonomy, IEEE International Conference on Robotics and Automation (ICRA 2010)*, 2010.

[6] A. Beyeler, S. Magnenat, and A. Habersaat, "Ishtar: a flexible and lightweight software for remote data access," *Proceedings of EMAV08, the 2008 European Micro Air Vehicle Conference*, 2008.

[7] F. Ruini and A. Cangelosi, "An evolutionary robotics 3d model for autonomous mavs navigation, target tracking and group coordination," *Proceedings of IJCNN 2010, International Joint Conference on Neural Networks*, pp. 760–767, 2010.

[8] F. Ruini and A. Cangelosi, "An incremental approach to the evolutionary design of autonomous controllers for micro-unmanned aerial vehicles," *Proceedings of TAROS 2010, 11th Conference Towards Autonomous Robotic Systems*, pp. 239–246, 2010.

[9] C. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," *Proceedings of ACM SIGGRAPH '87*, vol. 21, no. 4, pp. 25–34, 1987.

[10] A. Koestler and T. Braunl, "Mobile robot simulation with realistic error models," *Proceedings of the 2nd International Conference on Autonomous Robots and Agents*, pp. 46–51, 2004.

[11] P. Murray, "So what's new about complexity?," *Syst. Res.*, vol. 20, no. 5, pp. 409–417, 2003.

[12] C. Reynolds, "Steering behaviors for autonomous characters," *Proceedings of the Game Developers Conference*, pp. 763–782, 1999.

[13] C. Hartman and B. Benes, "Autonomous boids," *Computer Animation and Virtual Worlds*, vol. 17, pp. 199–206, 2006.

[14] H. Kunz and C. Hemelrjik, "Artificial fish schools: Collective effects of school size, body size and body form," *Artificial Life*, vol. 9, pp. 237–253, Jul 2003.

[15] J. Hodgins and D. Brogan, "Robot herds: Group behaviors for systems with significant dynamics," *Proceedings of ALIFE IV, the 4th International Workshop on the Synthesis and Simulation of Living Systems*, pp. 319–324, 1994.

[16] M. Nagy, Z. Akos, D. Biro, and T. Vicsek, "Hierarchical group dynamics in pigeon flocks," *Nature*, vol. 464, no. 7290, pp. 890–893, 2010.

[17] D. Biro, D. Sumpter, J. Meade, and T. Guilford, "From compromise to leadership in pigeon homing," *Current Biology*, no. 16, pp. 2123–2128, 2006.

[18] R. Freeman and D. Biro, "Modelling group navigation: Dominance and democracy in homing pigeons," *The Journal of Navigation*, no. 62, pp. 33–40, 2009.